

The background of the page is a light gray gradient. It is decorated with numerous 3D cubes of various colors (purple, blue, yellow, green, orange, pink, and gray) arranged in a scattered, overlapping pattern. Some cubes are solid, while others have a square hole in the center. Thin, curved lines in blue, red, and green connect some of the cubes, suggesting a network or flow. The title text is centered in the middle of the page.

Introdução à Programação Orientada a Objeto (online)

Cód.: TE 1756_0_EAD

Copyright © Impacta Participações e Empreendimentos Ltda.

Todos os direitos autorais reservados. Este material de estudo (textos, imagens, áudios e vídeos) não pode ser copiado, reproduzido, traduzido, baixado ou convertido em qualquer outra forma eletrônica, digital ou impressa, ou legível por qualquer meio, em parte ou no todo, sem a aprovação prévia, por escrito, da Impacta Participações e Empreendimentos Ltda., estando o contrafator sujeito a responder por crime de Violação de Direito Autoral, conforme o art.184 do Código Penal Brasileiro, além de responder por Perdas e Danos. Todos os logotipos e marcas utilizados neste material pertencem às suas respectivas empresas.

“As marcas registradas e os nomes comerciais citados nesta obra, mesmo que não sejam assim identificados, pertencem aos seus respectivos proprietários nos termos das leis, convenções e diretrizes nacionais e internacionais.”

Introdução à Programação Orientada a Objeto (online)

Coordenação Geral

Marcia M. Rosa

Coordenação Editorial

Henrique Thomaz Bruscagin

Supervisão de Desenvolvimento Digital

Alexandre Hideki Chicaoka

Produção, Gravação, Edição de Vídeo e Finalização

Rhomis Jonathan Moreira

Roteirização

José Eduardo Machado Grasso

Curso ministrado por

José Eduardo Machado Grasso

Edição e Revisão final

Fernanda Monteiro Laneri

Diagramação

Bruno de Oliveira Santos

Edição nº 1 | 1756/0_EAD
outubro/2015

Este material é uma nova obra derivada da seguinte obra original, produzida por TechnoEdition Editora Ltda., em Dez/2007: Introdução à Programação Orientada a Objeto
Autoria: Carla Raiter Paes, Eduardo José de Souza Engelmann, Henrique Thomaz Bruscagin e José Eduardo Machado Grasso
Nº de registro BN: 427925

Sobre o instrutor do curso:

José Eduardo Machado Grasso é analista, consultor e programador desde 1985, especializado em encontrar, adaptar ou criar tecnologias adequadas para projetos de TI de diversos segmentos. Desde 2007, vem trabalhando no ramo editorial, criando soluções para distribuição, leitura, transmissão segura e controle de distribuição de royalties em bibliotecas digitais. É instrutor da Impacta Tecnologia desde 2001, ministrando cursos de desenvolvimento de Software com plataforma .NET.

Apresentação	05
1. Apresentando a orientação a objetos	07
1.1. Introdução	08
1.1.1. Um pouco de história.....	08
1.2. Modelos orientados a objeto X modelos estruturados	09
1.3. Objetos.....	09
1.3.1. Objetos computacionais.....	11
1.4. Concepção de um sistema orientado a objeto	14
1.4.1. Análise.....	15
1.4.2. Programação.....	16
1.5. Vantagens.....	16
Teste seus conhecimentos.....	19
Mãos à obra!.....	23
2. Conceitos de orientação a objetos	25
2.1. Introdução	26
2.2. Objetos.....	26
2.2.1. Atributos	28
2.2.2. Operações e métodos	29
2.2.3. Mensagens.....	30
2.3. Classes	31
2.3.1. Instanciação.....	32
2.4. Herança	32
2.4.1. Herança simples	34
2.4.2. Herança múltipla.....	34
2.4.3. Classes abstratas	35
2.5. Persistência.....	35
2.6. Abstração	36
2.7. Encapsulamento.....	37
2.8. Polimorfismo	38
2.9. Compartilhamento	38
Teste seus conhecimentos.....	39
Mãos à obra!.....	43
3. Notações gráficas de classes e instâncias.....	45
3.1. Introdução	46
3.2. Modelo de objetos	47
3.2.1. Diagramas de classes.....	48
3.2.2. Diagramas de instâncias	50
Teste seus conhecimentos.....	51
Mãos à obra!.....	55

Introdução à Programação Orientada a Objeto (online)

4

4.	Estruturas e relacionamentos.....	59
4.1.	Introdução	60
4.2.	Generalização e herança	60
4.3.	Agregação	63
4.3.1.	Conexões entre objetos	64
4.3.1.1.	Conexão de ocorrência	64
4.3.1.2.	Conexão de mensagem.....	64
4.4.	Ligações e associações.....	65
	Teste seus conhecimentos.....	71
	Mãos à obra!.....	73
5.	Ambientes de desenvolvimento de software	73
5.1.	Introdução	74
5.2.	O que é um software.....	74
5.2.1.	Tipos de software	76
5.2.1.1.	Interface de usuário (User Interface ou UI).....	76
5.2.1.2.	Componentes	77
5.2.1.3.	Serviços	77
5.2.1.4.	Web services	78
5.3.	Linguagens de programação	78
5.4.	Bancos de dados	80
5.5.	Tecnologias e ferramentas	81
5.5.1.	Java.....	81
5.5.2.	Plataforma .NET	82
5.6.	Frameworks	82
5.7.	Metodologias de desenvolvimento	83
5.8.	Resumo	88
	Teste seus conhecimentos.....	89
	Mãos à obra!.....	93

Bem-vindo!

É um prazer tê-lo como aluno do nosso curso online de Introdução à Programação Orientada a Objeto. Este curso é ideal para você que pretende compreender os fundamentos da programação orientada a objetos, um modelo que é base para o desenvolvimento de programas em importantes linguagens.




No decorrer das aulas, você conhecerá a história do modelo de programação orientada a objeto, verá os conceitos de objeto, classe, herança, polimorfismo, encapsulamento, entre outros, estudará notações gráficas de classes e instâncias, o relacionamento entre objetos e os ambientes de desenvolvimento de software.

Para ter um bom aproveitamento deste curso, é imprescindível que você tenha participado do nosso curso de Introdução à Lógica de Programação ou possua conhecimentos equivalentes. Bom aprendizado!

Como estudar?



Este curso conta com:

-  Videoaulas sobre os assuntos que você precisa saber no curso de introdução à programação orientada a objeto.
-  Parte teórica, com mais exemplos e detalhes para você que quer se aprofundar no assunto da videoaula.
-  Exercícios de testes e laboratórios para você pôr à prova o que aprendeu.



Apresentando a orientação a objetos 1

- ✓ Definição de programação orientada a objetos;
- ✓ Comparação entre modelos orientados a objeto e modelos estruturados;
- ✓ Definição de objetos;
- ✓ Conceito de sistema orientado a objetos;
- ✓ Vantagens da orientação a objetos.

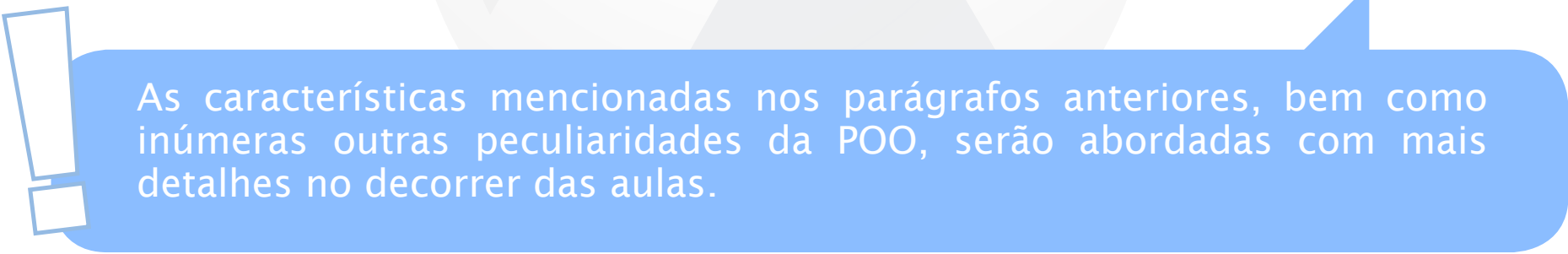
1.1. Introdução

Originada em meados da década de 1960 – mas utilizada em larga escala apenas a partir do início dos anos 1990 –, a linguagem orientada a objetos (POO) é um modelo de programação que emprega objetos no desenvolvimento de aplicações. Ela é baseada em vários conceitos, como modularidade, herança, encapsulamento e polimorfismo, os quais são amplamente aplicados na etapa de desenvolvimento.

Em comparação à visão tradicional de programação, em que o programa pode ser visto como uma lista de instruções para o computador, na POO, cada um dos objetos relaciona-se com os demais. Essa habilidade é demonstrada em diferentes características, como:

- Processamento de dados;
- Recebimento de mensagens de outros objetos;
- Envio de mensagens para outros objetos da aplicação.

Uma das vantagens de utilizar a programação orientada a objetos é que ela permite alterar ou substituir partes de um sistema sem que haja riscos de ocorrência de erros. Além disso, ela oferece uma visão mais natural, uma forte arquitetura e um alto grau de reusabilidade do código.



As características mencionadas nos parágrafos anteriores, bem como inúmeras outras peculiaridades da POO, serão abordadas com mais detalhes no decorrer das aulas.

1.1.1. Um pouco de história

Resgatando um pouco a respeito da origem da POO, especialmente os fatores que conduziram ao seu surgimento na década de 1960, constata-se um período em que o software e o hardware tornaram-se bastante complexos.

A qualidade do software deveria ser mantida a qualquer custo – isso era o que almejavam os profissionais da época. Justamente nesse contexto, em que se chegou a discutir até mesmo a ideia de uma crise relacionada ao software, surgiu a POO. Ela pretendia combater o problema, principalmente por meio da modularidade em software, ou seja, unidades separadas de lógica de programação.

A primeira linguagem de programação que introduziu os conceitos que fundamentam a POO foi a linguagem **Simula**. Esses conceitos, que incluem objetos, classes, subclasses, métodos virtuais, corrotinas, garbage collection e simulação de eventos discretos, foram introduzidos como um superconjunto da linguagem Algol (ALGOritmic Language).

A primeira linguagem de programação a ser chamada de orientada a objetos foi a linguagem **Smalltalk**.

1.2. Modelos orientados a objeto X modelos estruturados

A orientação a objetos e a programação estruturada são modelos diferentes de desenvolvimento de sistemas. O conceito de programação estruturada diz respeito a uma estratégia baseada na ideia de que um sistema é dividido em duas partes principais: os dados e as funcionalidades.

Em projetos estruturados, geralmente os dados são representados por um modelo de dados que define os registros estruturados para serem armazenados nas bases de dados. As funcionalidades são definidas por um modelo de processamento de fluxo de dados no qual é determinado como os dados devem entrar e sair dos processos e como devem ser armazenados no sistema.

Portanto, nas aplicações que utilizam uma abordagem estruturada, os dados ficam separados das funcionalidades do sistema. Já em um sistema de software que se baseia no conceito de orientação a objetos, os dados ficam armazenados em cada objeto existente. Em termos simples, objeto é algo capaz de ser apresentado e que possui algum significado ou sentido.

Em vez de os sistemas orientados a objetos serem definidos estruturalmente como duas partes separadas, as funcionalidades são nada mais que conjuntos de objetos interativos. Esses objetos geralmente possuem ações que interferem no comportamento de um sistema e informações que armazenam dados. Em termos de modelagem, os objetos são representados por classes.

1.3. Objetos

A orientação a objetos visa representar, de maneira idêntica, as situações do mundo real nos sistemas computacionais. Para ela, os sistemas operacionais não devem ser vistos como uma coleção estruturada de processos, mas sim como uma coleção de objetos que interagem uns com os outros organizadamente, assim como ocorre no mundo real.

Em uma concepção abrangente, um objeto pode ser entendido como sendo um elemento físico, por exemplo, uma pedra, uma cadeira ou um animal, ou como um elemento abstrato, como uma conta bancária.

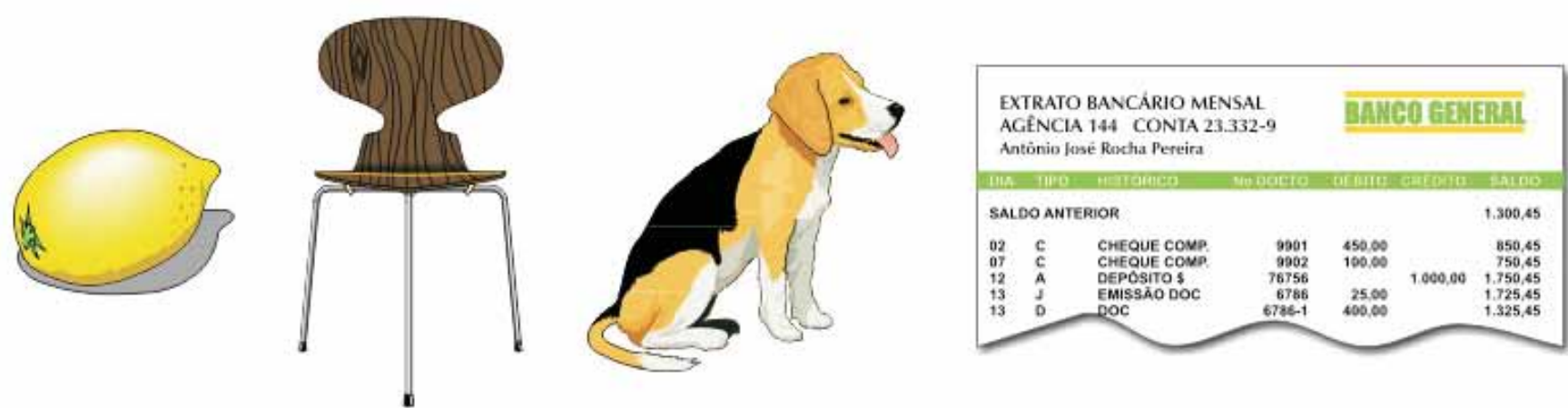


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Dentro do ambiente de TI, os objetos computacionais são objetos que estão localizados dentro de sistemas de computador. A intenção ao especificar esses objetos é retratar as mesmas características dos objetos do mundo real e programar comportamentos que se aproximem dos encontrados na realidade. Como o comportamento desses objetos já está programado, um programador não precisa entender o funcionamento interno desse comportamento para interagir com ele. Para isso, basta ativar os comportamentos programados.

Um exemplo comum desse processo são os aplicativos para desenvolvimento de softwares que utilizam recursos de arrastar e soltar. O programador pode criar uma tela arrastando e soltando botões, caixas de textos, listas, conexões com banco de dados para uma tela, sem se preocupar com a forma como esses objetos funcionam internamente.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Ao arrastar um botão, o programador espera que o software, ao ser executado, desenhe aquele botão, e que este, ao ser pressionado, funcione como um botão do mundo real, afundando dentro do painel no qual está contido e executando alguma ação. O botão é um objeto e o fato dele responder a um clique é um evento. Esse tipo de programação em que objetos são desenhados por um aplicativo e o programador escreve as ações a serem executadas ficou conhecido como **Programação Orientada a Eventos**.

1.3.1. Objetos computacionais

A representação dos objetos computacionais no computador retrata apenas uma imagem dos objetos do mundo real, ou seja, são modelos abstratos dos objetos reais. No entanto, o comportamento desses modelos pode ser igual ao da realidade. Para que ocorra a interação dentro de um sistema de computador, os objetos computacionais devem saber como reagir diante de uma ação.

Conforme mencionado anteriormente, o comportamento desses objetos já está previamente programado, portanto, eles possuem a informação necessária para saber como reagir. Isso significa que o seu comportamento não depende do sistema.

Para facilitar a compreensão acerca da relação entre os objetos computacionais e o comportamento associado a eles, consideremos, como exemplo, quatro tipos de objetos existentes no ambiente computacional:

- **Objetos visuais**

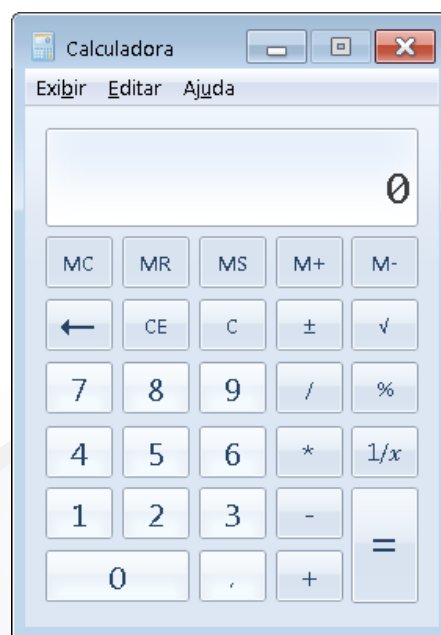
Interagem diretamente com o usuário, permitindo que ele aperte botões, mova peças e selecione itens dentro de caixas. Neste caso, a interação do usuário com esses objetos ocorre em um monitor, com a utilização de um mouse ou teclado. Mesmo assim, existe um nível de abstração e interação muito bom em relação ao mundo real.

Em geral, esse tipo de objeto é manuseado por ambientes de programação visual. Utilizar os objetos desses ambientes é uma tarefa simples: basta selecioná-los em uma palheta e, em seguida, arrastá-los até a janela da aplicação.

Um fator importante e que exige a nossa máxima atenção é a maneira como utilizaremos esses objetos, pois cada objeto já possui o conhecimento de como deve funcionar, sem que o programador escreva um código.

Alguns exemplos de objetos visuais são: Menus, Caixas de Texto, Botões e Listas.

A calculadora do Windows utiliza diversos objetos visuais:



É possível identificar nessa imagem da calculadora: um menu, uma caixa de texto e diversos botões.

- **Objetos de domínio de trabalho**

São objetos que estão envolvidos ou são criados durante o desenvolvimento de um sistema de computador. Além disso, eles se relacionam diretamente como trabalho desenvolvido e, muitas vezes, não são percebidos pelos olhos do usuário final.

Outra característica dos objetos de domínio de trabalho é que eles estão envolvidos diretamente com o sistema de informações.

Por exemplo, um software controlador de estoque pode conter alguns objetos internos, como Cliente, Produto, Fornecedor, Venda ou Compra.

- **Objetos com tarefa relacionada**

São arquivos que também possuem comportamentos inclusos, como objetos do tipo documentos e arquivos multimídia. Apesar de apresentarem esses comportamentos, em muitos casos é necessário ter softwares aplicativos para abri-los.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Por exemplo, um software de compra de ingressos para o cinema pode ter uma funcionalidade que permite ao usuário ver o trailer de um filme. Se esse trailer for um arquivo do tipo **MOV**, um aplicativo chamado **QuickTime**, da Apple, deverá estar instalado no computador, além do aplicativo que é responsável pela exibição correta de imagem e som.

- **Objetos multimídia**

A maioria das pessoas já está familiarizada com os objetos multimídia, os quais podem conter som, imagem, animação ou vídeo. Esses objetos sabem de que forma devem se comportar diante de uma situação.

Por representarem os objetos reais de forma bastante idêntica, os objetos multimídia são de fácil utilização.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

1.4. Concepção de um sistema orientado a objeto

O conceito de orientação a objetos consiste em criar um sistema computacional como um sistema orgânico, que seja formado por objetos que interajam uns com os outros.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Esse conceito pode ser empregado na análise de sistemas e na programação, fazendo com que uma das principais vantagens da orientação a objetos seja a possibilidade de utilizar a mesma técnica, tanto para a definição lógica do sistema quanto para a sua implementação.

1.4.1. Análise

A análise orientada a objetos consiste em definir os objetos que pertencem a um sistema e a maneira como eles se comportam. O processo dessa análise consiste em identificar os seguintes elementos:

- Os objetos que existem dentro do ambiente que desejamos automatizar;
- Os atributos desses objetos, ou seja, que tipos de informação esses objetos devem conter;
- As ações que esses objetos podem executar.

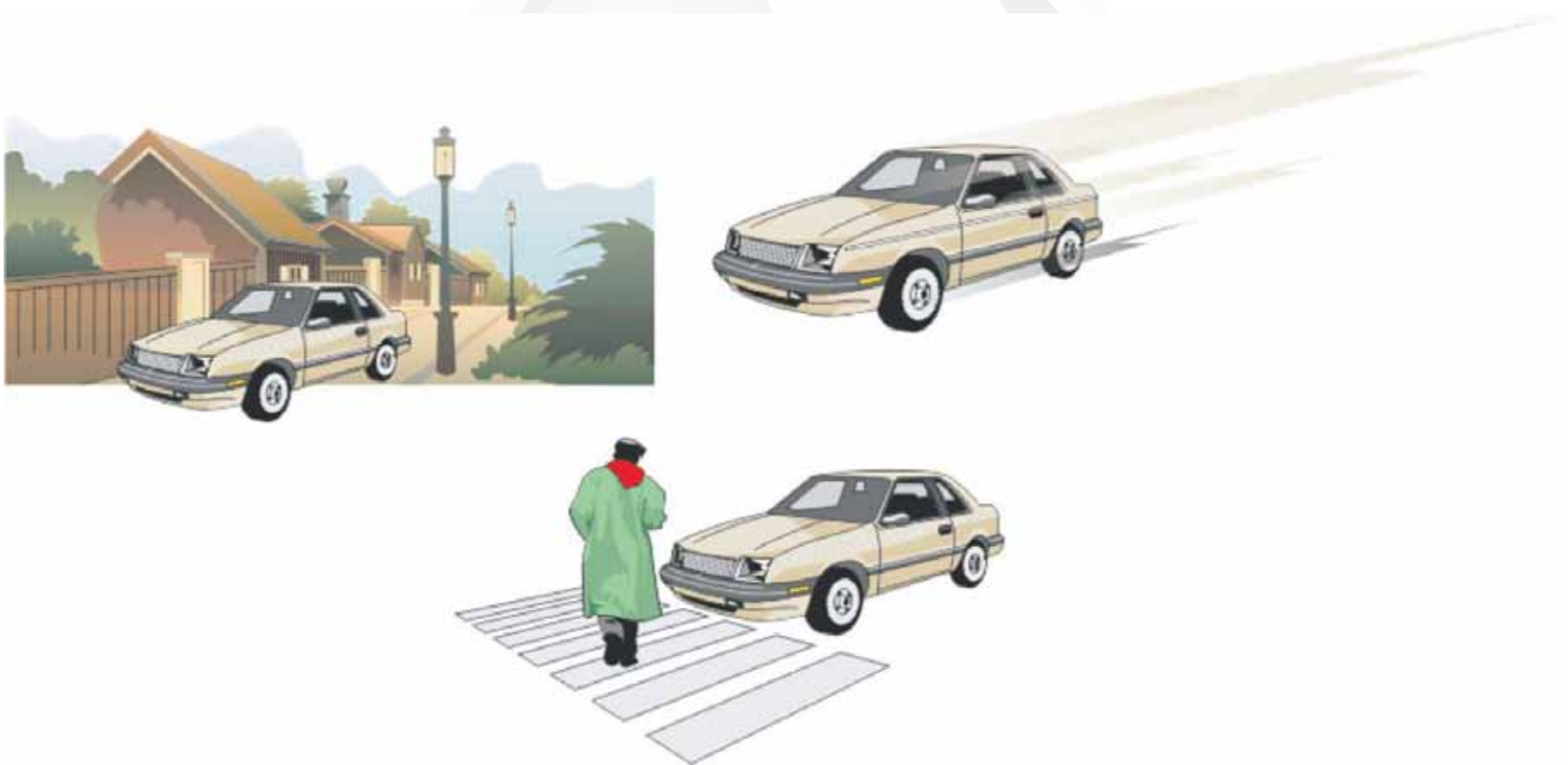


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

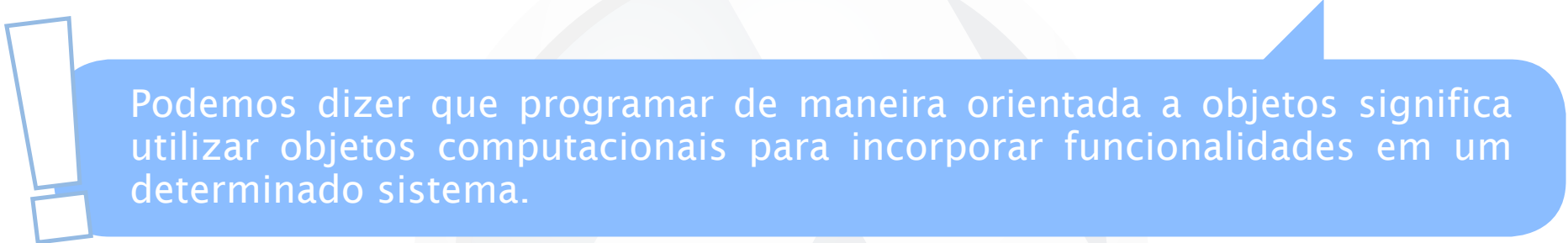
Por meio da análise orientada a objetos, um analista pode compartilhar o conhecimento que ele tem em relação ao funcionamento do sistema com um usuário comum.

1.4.2. Programação

Consiste em utilizar estruturas de dados que façam uma simulação do comportamento dos objetos. Dessa forma, a programação é realizada com mais facilidade com a utilização de uma única técnica tanto para a análise quanto para a programação.

Na programação orientada a objetos, os objetos identificados pelo analista em seu diagrama podem ser implementados exatamente da forma que foram projetados. Por outro lado, na análise estruturada é necessário traduzir os diagramas da análise para estruturas de dados e de programação.

Depois de fazer uma análise orientada a objetos, é possível implementá-la em uma linguagem tradicional e, também, implementar sistemas analisados com outras técnicas, utilizando linguagens orientadas a objetos.



Podemos dizer que programar de maneira orientada a objetos significa utilizar objetos computacionais para incorporar funcionalidades em um determinado sistema.

1.5. Vantagens

A seguir veremos algumas vantagens da utilização da técnica de orientação a objetos:

- **Organização**

A partir do momento que os desenvolvedores de sistemas começam a utilizar a técnica da orientação a objetos, eles passam a usufruir de diversas vantagens, dentre as quais a principal é a possibilidade de reunir em uma mesma estrutura os dados e os processos que são executados sobre esses dados. Consequentemente, há um aumento no nível de organização e simplicidade do programa.

- **Produtividade**

Outra vantagem é o ganho de produtividade, pois, quando um desenvolvedor cria um objeto para realizar uma tarefa difícil ou altera a maneira como esse objeto a executa internamente, os demais desenvolvedores que venham a interagir com esse objeto precisarão apenas acessá-lo para executar as mesmas tarefas, sem a necessidade de saber de que forma ocorre a execução e sem precisar alterar seus programas para continuar tendo acesso ao novo comportamento.

- **Redução do custo**

Além das vantagens mencionadas anteriormente, também há a redução no custo de desenvolvimento e no risco de ocorrência de erros. Isso ocorre porque, depois de criar uma estrutura eficaz de objetos, será possível incluir módulos adicionais no sistema, que reutilizem funcionalidades já desenvolvidas, ou seja, utilizem o mesmo código em sistemas diferentes. Consequentemente, não há necessidade de reprogramação.

- **Reaproveitamento**

Mais uma vantagem encontrada é a possibilidade de transformar objetos semelhantes, com a finalidade de aproveitá-los em novos sistemas. Isso significa que podemos fazer pequenas alterações em objetos que possuem características parecidas com as especificações necessárias de um novo objeto, a fim de que esse novo objeto resultante da alteração seja utilizado em um novo sistema.

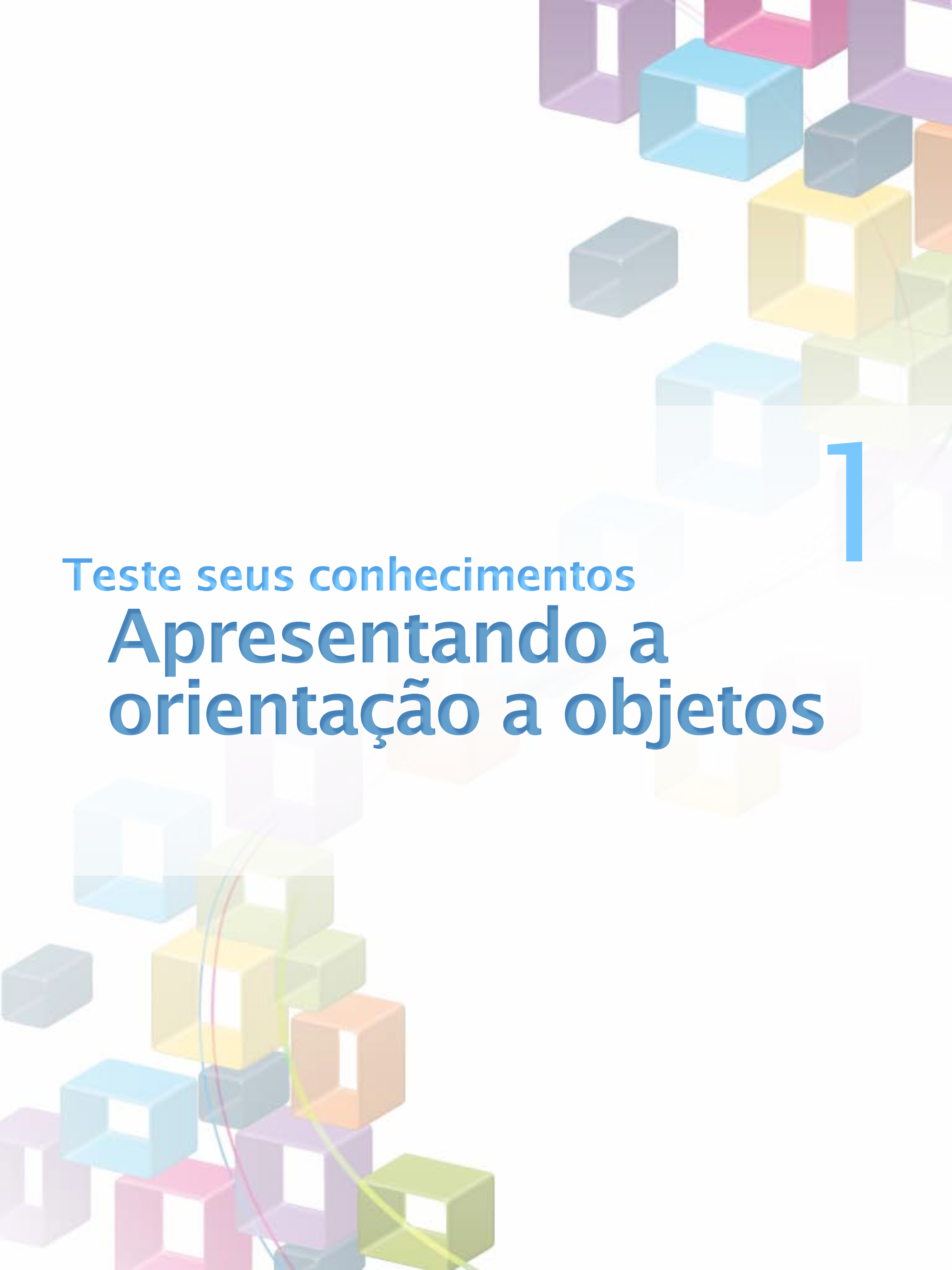
- **Facilidade de manutenção**

Quando é preciso realizar alguma alteração em um sistema, a flexibilidade oferecida pela técnica de orientação a objetos permite que o desenvolvedor adapte, exclua ou inclua novos objetos no sistema rapidamente, sem comprometer o funcionamento do mesmo.

- **Trabalho em equipe**

Usando orientação a objetos, é fácil dividir tarefas entre diversas equipes de programação. Uma parte da equipe pode trabalhar no código relacionado aos bancos de dados e outra parte pode trabalhar na interação com o usuário. Esse tipo de programação é conhecido como **Programação em Camadas**.

Enfim, como a programação orientada a objetos lida com objetos pré-prontos, ela não apenas simplifica a criação de novos sistemas como tem a finalidade de maximizar a reutilização dos objetos.

The background features a collection of 3D cubes in various colors (purple, blue, yellow, green, orange, pink, grey) arranged in a scattered, overlapping manner. A large, bold blue number '1' is positioned on the right side of the image. The text is centered in the lower half of the image.

Teste seus conhecimentos Apresentando a orientação a objetos

1. Qual dos itens a seguir não representa uma vantagem ao se adotar um modelo de programação orientada a objetos?

- ☐ a) Produtividade
- ☐ b) Organização
- ☐ c) Reaproveitamento de código
- ☐ d) Gravação mais rápida no banco de dados
- ☐ e) Facilidade de manutenção

2. Qual dos itens a seguir não é um exemplo de objeto?

- ☐ a) Uma nota fiscal
- ☐ b) Uma caneta
- ☐ c) A cor do céu
- ☐ d) Um boleto bancário
- ☐ e) Um DVD

3. Que tipo de objeto não é percebido diretamente por um usuário de software?

- ☐ a) Objeto de domínio de trabalho
- ☐ b) Objeto visual
- ☐ c) Objeto multimídia
- ☐ d) Objeto com tarefa relacionada
- ☐ e) Nenhuma das alternativas anteriores está correta.

4. Definir os objetos que pertencem a um sistema e o seu comportamento compõe qual parte do ciclo de desenvolvimento de software?

- ☐ a) Implantação
- ☐ b) Análise
- ☐ c) Testes
- ☐ d) Venda
- ☐ e) Design

5. Como é conhecido o tipo de programação no qual uma parte do programa realiza as tarefas de gravação no banco, outra define as regras do negócio e outra define a interação com o usuário?

- ☐ a) Programação em bloco
- ☐ b) Programação em camadas
- ☐ c) Programação visual
- ☐ d) Programação orientada a dados
- ☐ e) Nenhuma das alternativas anteriores está correta.

The background features a collection of 3D cubes in various colors (purple, blue, yellow, green, orange, pink, grey) arranged in a scattered, overlapping manner. Some cubes are solid, while others have a square hole in the center. Thin, curved lines in blue, red, and yellow weave through the cubes, adding a sense of movement and complexity to the composition.

Mãos à obra!

Apresentando a orientação a objetos

Laboratório 1

Para este exercício não existe uma única resposta certa e definitiva.

A – Definindo objetos para construir aplicações

1. Leia os problemas propostos a seguir e defina pelo menos cinco objetos que seriam necessários para construir as aplicações:

Situação 1

A empresa X é especializada na venda de equipamentos para pesca e necessita de um sistema para controlar o estoque, calcular comissões, pagar em dia os fornecedores e funcionários e gerar relatórios de vendas para os seus acionistas. As vendas são feitas pela Internet ou pelas diversas lojas espalhadas pelo Brasil, e os clientes podem pagar com boleto bancário, cartão de crédito, cheque ou dinheiro. Cada vendedor recebe 5% de comissão sobre as vendas.

Situação 2

A academia de ginástica X necessita de um sistema para controlar o agendamento de aulas, recebimento das mensalidades dos alunos e o pagamento dos professores e funcionários. Existem diversos planos de condicionamento físico. Alguns são individuais, como a musculação, e outros em turmas, como Yoga e Natação. As mensalidades variam de acordo com o curso e o número de aulas por semana.

A decorative background featuring a large, semi-transparent number '2' on the right side. The background is filled with numerous 3D blocks of various colors (purple, blue, yellow, green, orange, pink, grey) arranged in a scattered, overlapping manner, some of which are hollow cubes. The overall aesthetic is modern and geometric.

2

Conceitos de orientação a objetos

- ✓ Objetos;
- ✓ Classes;
- ✓ Herança;
- ✓ Persistência;
- ✓ Abstração;
- ✓ Encapsulamento;
- ✓ Polimorfismo;
- ✓ Compartilhamento.

2.1. Introdução

A abordagem de organização proposta pelo termo orientação a objetos é muito diferente da abordagem presente no desenvolvimento tradicional de um software, em que as rotinas e as estruturas de dados são desenvolvidas de maneira linear e sem relacionamento direto entre as ações e dados.

Alguns conceitos considerados fundamentais no desenvolvimento de bons programas, como a abstração e a encapsulação, são favorecidos pela abordagem de orientação a objetos. Esses conceitos não são, porém, exclusivos dessa abordagem, mas apenas melhor suportados nesta do que em outras metodologias.

2.2. Objetos

Para entendermos melhor o conceito de objeto, podemos considerá-lo como uma representação do mundo real. Em termos gerais, entendemos por objetos quaisquer elementos da natureza aos quais é possível atribuir comportamentos e características. No entanto, em termos de computação, entendemos por objetos os elementos capazes de representar uma entidade que esteja no domínio de interesse do problema analisado. As entidades representadas por objetos podem ser concretas ou abstratas.

As linguagens de programação que são orientadas a objetos possuem mecanismos que, a partir do conceito de classes, permitem determinar os tipos de objetos que serão destinados a cada aplicação. Os objetos que apresentam semelhanças entre si são agrupados em classes.



Os objetos, quando são criados, ocupam um espaço na memória que é utilizado para o armazenamento de seu estado e de um grupo de operações, as quais podemos aplicar ao objeto. Mais especificamente, o estado de um objeto refere-se aos valores de seu conjunto de atributos, e seu grupo de operações refere-se ao conjunto de métodos. A classe é responsável por definir ambos, atributos e métodos.

Tendo em vista que os objetos são instâncias de classe, para que eles realizem suas tarefas é preciso que as instâncias sejam criadas, uma vez que, por meio de sua manipulação, as tarefas dos objetos podem ser efetuadas. Uma vez realizadas essas tarefas, podemos excluir os objetos.

Ainda em relação aos objetos, é importante notar alguns aspectos:

- Todos os objetos são distinguíveis e possuem uma identidade própria;
- Os objetos possuem duas finalidades: promover o entendimento do mundo real e dar suporte a uma base prática para uma implementação computacional;
- A decomposição de um problema em objetos depende da natureza do problema e do julgamento do projetista, ou seja, não existe uma maneira correta ou única de fazer esta decomposição;
- Um objeto é constituído por atributos e métodos.

Podemos citar como exemplos de objetos: Nota Fiscal, Cliente, Produto, Boleto, Orçamento.

2.2.1. Atributos

Os valores de dados assumidos pelos objetos de uma classe são chamados de atributos. Para cada instância do objeto, um atributo possui um valor, que pode ser o mesmo para instâncias diferentes. Para exemplificar, o objeto **Pessoa** pode possuir os atributos **Nome** e **Altura**, e o objeto **Produto** pode possuir os atributos **Nome**, **Preço** e **EstoqueAtual**, ou seja, os atributos podem ser entendidos como variáveis ou campos utilizados para armazenar os diferentes valores que as características dos objetos podem conter.

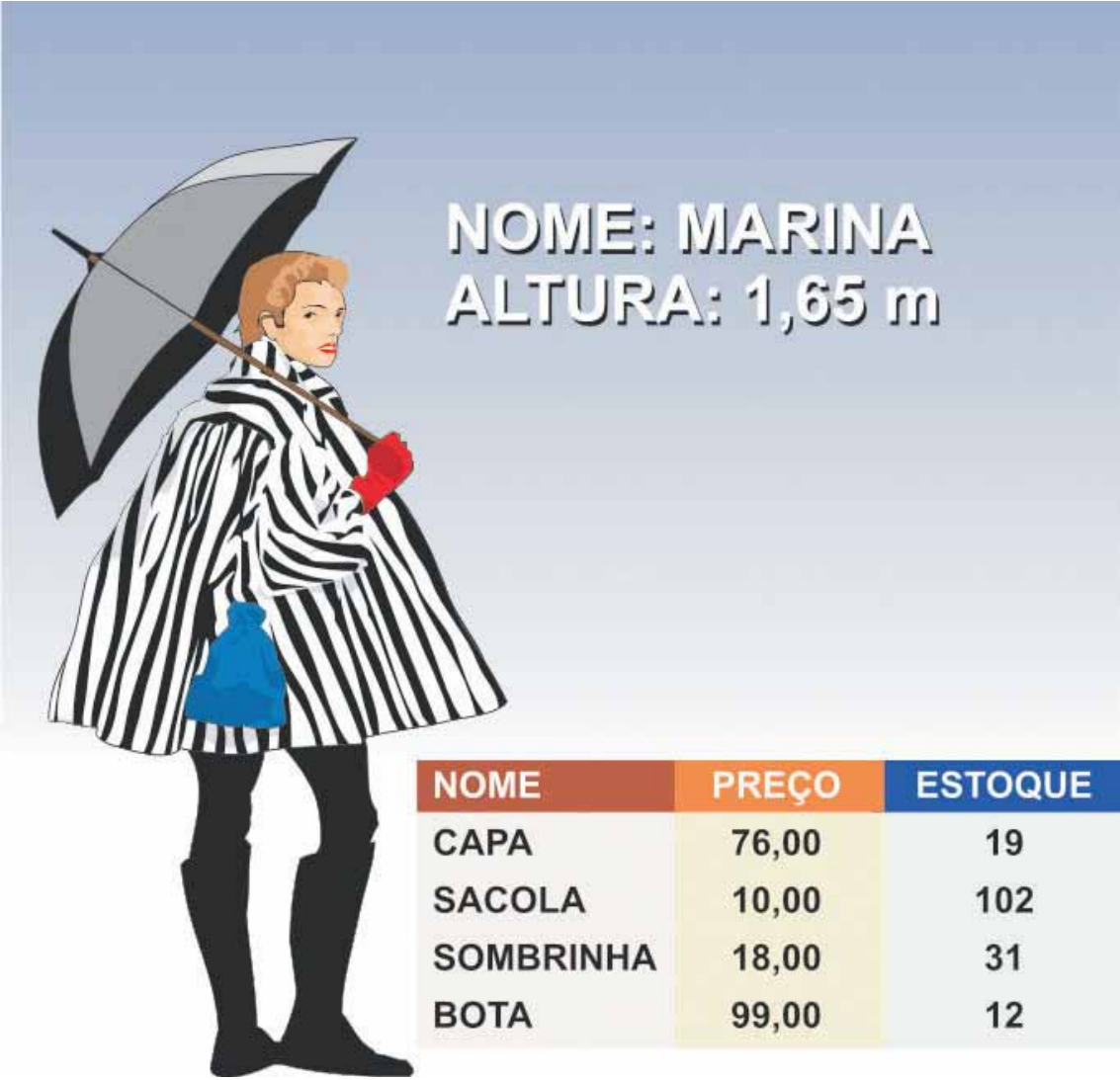


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Dentro de uma classe, os nomes de cada atributo devem ser exclusivos, porém, podemos ter atributos com o mesmo nome em classes diferentes.

Os atributos dos objetos só podem ter seus valores alterados por meio de estímulos internos ou externos. Para modificar esses valores, é necessário disparar eventos que provoquem a transição de estados no objeto.

Os dois itens a seguir garantem a independência completa de qualquer objeto em relação aos demais, além da possibilidade de alteração dos atributos e do código interno dos métodos do objeto, sem o risco de afetar outros objetos internamente:

- Cada objeto é responsável pela mudança de seus próprios atributos;
- Salvo por meio da solicitação de serviços, nenhum objeto possui a capacidade de interferir nos atributos de outro objeto.

2.2.2. Operações e métodos

Vejamos o que são as operações e os métodos:

• Operações

Operação é uma transformação, ou função, que pode ser aplicada a objetos de uma classe ou por esses objetos. Dentro de uma classe, as mesmas operações são compartilhadas por todos os objetos.

Toda operação possui, como argumento implícito, um objeto-alvo, cuja classe determinará o comportamento da operação. O objeto conhece a classe à qual pertence e por isso é possível escolher a implementação correta da operação.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Operações que se aplicam a diversas classes diferentes são chamadas de polimórficas, pois podem assumir formas diferentes em cada uma das classes.

- **Métodos**

Os métodos permitem que um objeto se manifeste e interaja com outros objetos. Método é uma implementação de uma operação para uma classe específica.

Um método possui uma assinatura constituída pelo nome de uma operação, o número, tipo e ordem de seus argumentos e pelo valor de retorno. Como estratégia de desenvolvimento, é recomendável manter, além de um comportamento consistente entre as implementações, assinaturas que sejam coerentes para os métodos que implementem uma determinada operação.

Como exemplo, podemos citar que o objeto **Conta** pode conter os métodos **Retirar** e **Depositar**.

2.2.3. Mensagens

A mensagem, ou seja, o meio de comunicação entre os objetos, é uma chamada feita a um objeto com o objetivo de invocar um de seus métodos, ativando um comportamento descrito por sua classe. A mensagem é uma requisição de ação junto com argumentos necessários para a execução da tarefa solicitada.



Vejamos um exemplo:

MinhaConta.Depositar(100)

O objeto **MinhaConta** está enviando a mensagem **Depositar** e passando o parâmetro **100**. Esse valor será usado para executar a ação corretamente (depositar 100 reais na conta representada pelo objeto (**MinhaConta**)).

2.3. Classes

As classes criam representações computacionais a partir de entidades do mundo real. São elementos fundamentais no desenvolvimento de softwares orientados a objetos e podem ser definidas como descrições coletivas ou genéricas do mundo real. Assim, em um sistema, a definição das classes deve procurar inspiração nas entidades mundanas.

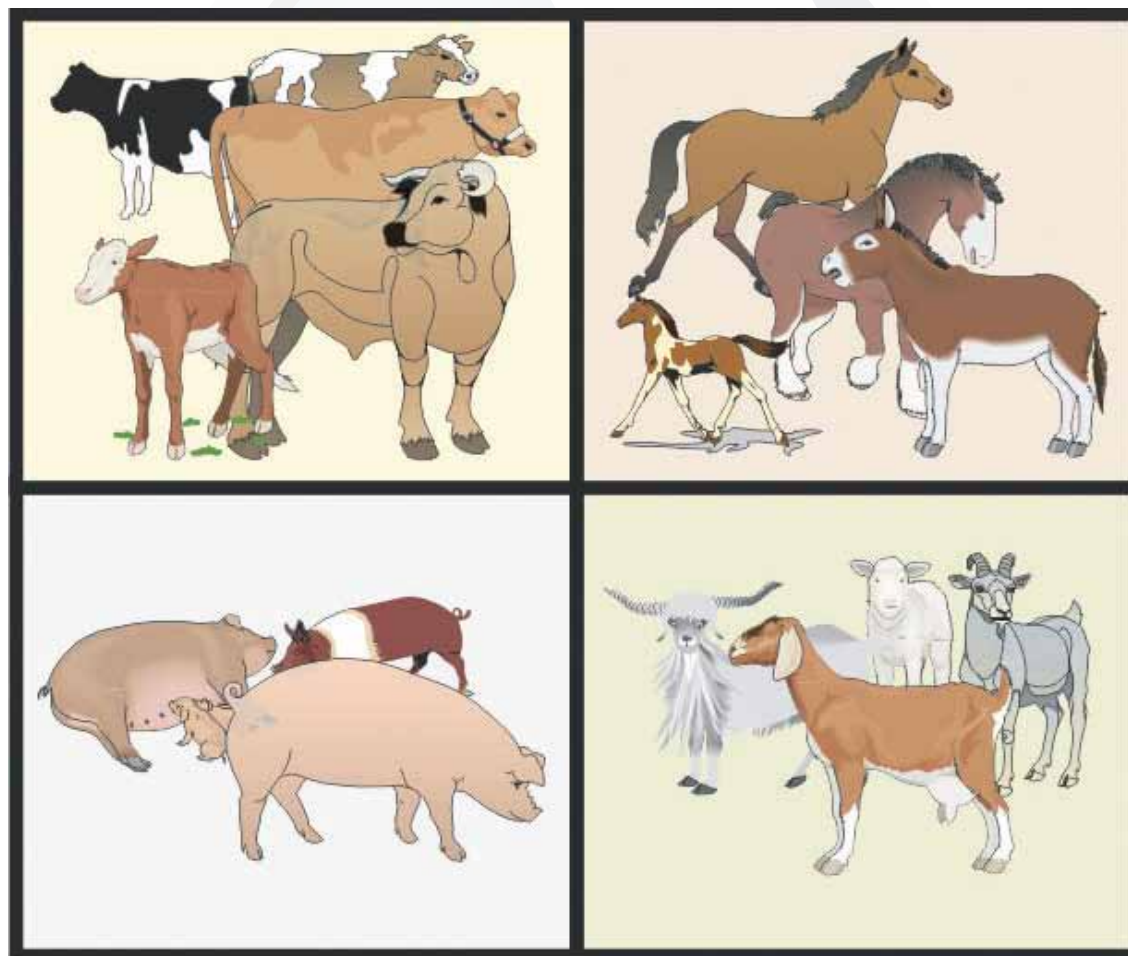


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Computacionalmente, podemos definir classe como uma abstração de um conjunto de objetos, os quais são agrupados por possuírem similaridades em termos de comportamento (operações) e características (atributos). Sendo assim, as propriedades ou os atributos de um objeto são descritos a partir da definição de uma classe.

É importante notar que os objetos são criados pela classe, ou seja, não criamos os objetos, mas sim definimos, na classe, os atributos e métodos necessários para essa criação.

O ato de criar um objeto é chamado de **Instanciação**. Instanciar um objeto é criar uma cópia de uma classe na memória, para uso no programa.

2.3.1. Instanciação

Na teoria de orientação a objetos, um objeto é definido como uma instância de uma classe. A instanciação é a criação de um objeto por uma classe, em que esta funciona como um gabarito, ou modelo, para essa criação.

2.4. Herança

A herança possibilita que as classes compartilhem seus atributos, métodos e outros membros da classe entre si. Para a ligação entre as classes, a herança adota um relacionamento esquematizado hierarquicamente.

O conceito relacionado ao mecanismo de herança é um dos maiores diferenciais entre a programação orientada a objetos e os outros tipos de programação. Por meio da herança é possível estender as definições existentes. Quando temos uma hierarquia de classes planejada de forma adequada, temos a base para que um código possa ser utilizado novamente, o que permite poupar esforço e tempo, na medida em que o desenvolvimento de um código exige ambos de forma considerável.

Na herança, temos dois tipos principais de classe:

- **Classe base:** A classe que concede as características a uma outra classe;
- **Classe derivada:** A classe que herda as características da classe base.

O fato de as classes derivadas herdarem atributos das classes base assegura que programas orientados a objetos cresçam de forma linear, e não geométrica, em complexidade.

Cada nova classe derivada não possui interações imprevisíveis em relação ao restante do código do sistema.

Com o uso da herança, uma classe derivada geralmente é uma implementação específica de um caso mais geral. A classe derivada deve apenas definir as características que a tornam única.

Por exemplo, uma classe base, que serviria como um modelo genérico, poderia ser a classe **Produto** com os campos **Nome** e **Preço**. Já uma classe derivada poderia ser a **Livro**, com os campos **Nome** e **Preço** herdados de **Produto** e acrescida do campo **Numero de Páginas**.

De maneira natural, as pessoas visualizam o mundo como sendo formado de objetos relacionados entre si hierarquicamente. Vejamos a seguir a relação entre animais, mamíferos e cachorros.

Os animais, sob uma descrição abstrata, apresentam atributos, tais como tamanho, inteligência e estrutura óssea, além de aspectos comportamentais, como mover-se, dormir, respirar, alimentar-se, entre outros. Os atributos e aspectos comportamentais descritos definem a classe dos animais.

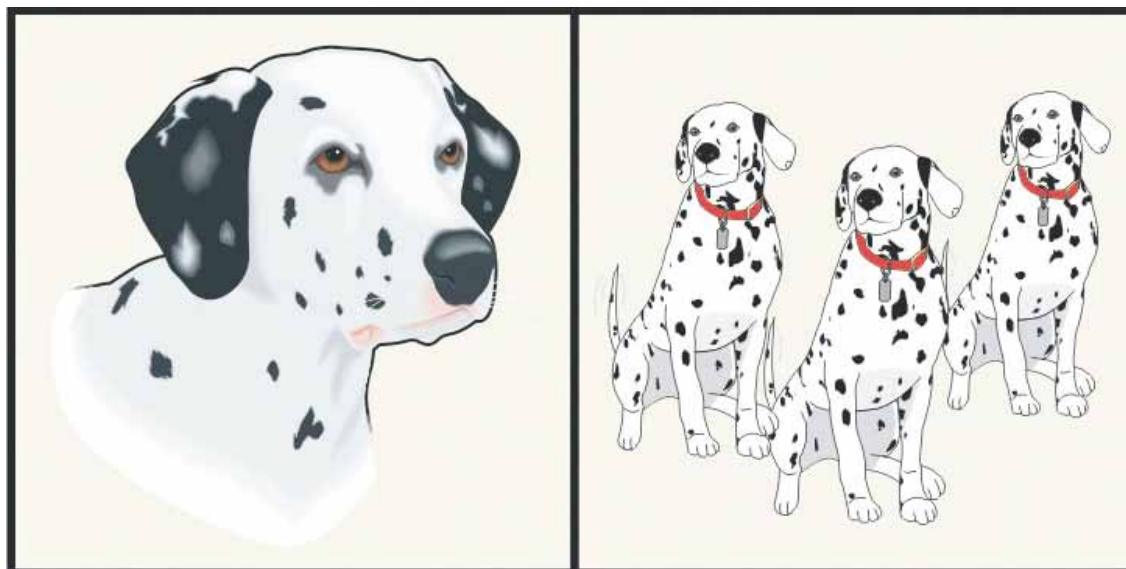


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Se analisarmos os mamíferos, que estão dentro da classe de animais, notaremos atributos mais particulares, como tipo de dente, pelos e glândulas mamárias. Os mamíferos são classificados como uma classe derivada dos animais, que por sua vez são uma classe base de mamíferos.

Pela chamada hierárquica de classes, a classe derivada mamíferos recebe todos os atributos de animais, partindo do princípio que uma classe derivada recebe por herança todos os atributos de seus ancestrais.

No exemplo seguinte, temos a hierarquia de classes do cachorro:

Reino	Animal
Filo	Cordata (cão, lobo, raposa, gato, cavalo, cobra, sapo, peixe)
Classe	Mammalia (cão, lobo, raposa, gato, cavalo)
Ordem	Carnívora (cão, lobo, raposa, gato)
Família	Canídea (cão, lobo, raposa)
Gênero	Canis (cão, lobo)
Espécie	Canis familiares (cão): dálmata, pastor, dobermann etc.

Adotemos como exemplo um cão da raça **dálmata**, pertencente à espécie **Canis familiares**, como visto no exemplo anterior. Por herança, o dálmata herda as características do gênero **Canis**, que por sua vez herda as características da família **Canídea**, que por sua vez herda as características da ordem **Carnívora**, e assim sucessivamente.


2.4.1. Herança simples

A herança simples determina que uma classe herdará características de apenas uma superclasse.

2.4.2. Herança múltipla

A herança múltipla determina que uma classe herdará características de duas ou mais superclasses, como é o caso das classes derivadas, quando herdarem duas ou mais classes base.

Quando utilizamos a herança múltipla, é importante observarmos a manipulação de nomes de membros duplicados nas classes base. Para especificar a qual membro a classe derivada se refere, utilizamos o mecanismo chamado qualificação. A qualificação consiste na prefixação do nome do membro com o nome da classe base a que ele faz referência. Os nomes do membro (atributo ou método) e da classe base são separados por um duplo dois-pontos (::).



Existem muitas discussões sobre a questão da utilização da herança múltipla e suas vantagens. Um ponto importante a ressaltar é que os exemplos para ilustrar a utilidade da herança múltipla, em muitos casos, são pouco naturais. Contudo, não se coloca em questão o mecanismo de herança múltipla em si, mas sim a sua validade como ferramenta de modelagem de aplicações. Sua utilização acaba sendo uma questão de escolha pessoal, visto que em diversas linguagens o mecanismo está presente.

2.4.3. Classes abstratas

Entendemos por classes abstratas as classes a partir das quais não é possível realizar qualquer tipo de instância. São classes feitas especialmente para serem modelos para suas classes derivadas. As classes derivadas, via de regra, deverão sobrescrever os métodos para realizar a implementação dos mesmos. As classes derivadas das classes abstratas são conhecidas como **Classes Concretas**.

Como medidas de segurança, as classes abstratas podem ser somente estendidas e a criação de um objeto a partir da mesma é um procedimento evitado. Além disso, caso um ou mais métodos abstratos estejam presentes nessa classe abstrata, a classe filha será, então, forçada a definir tais métodos, pois, caso contrário, a classe filha também se tornará abstrata.

A funcionalidade dos métodos abstratos que são herdados pelas classes filha normalmente é atribuída de acordo com o objetivo ou o propósito dessas classes. É possível, porém, não atribuirmos uma funcionalidade a esses métodos abstratos. Nesse caso, faz-se necessário, pelo menos, declará-los.

Os métodos abstratos estão presentes somente em classes abstratas, e são aqueles que não possuem implementação.

2.5. Persistência

A persistência de um objeto é o tempo que este permanece armazenado na memória RAM (principal) ou auxiliar (um meio magnético, por exemplo).

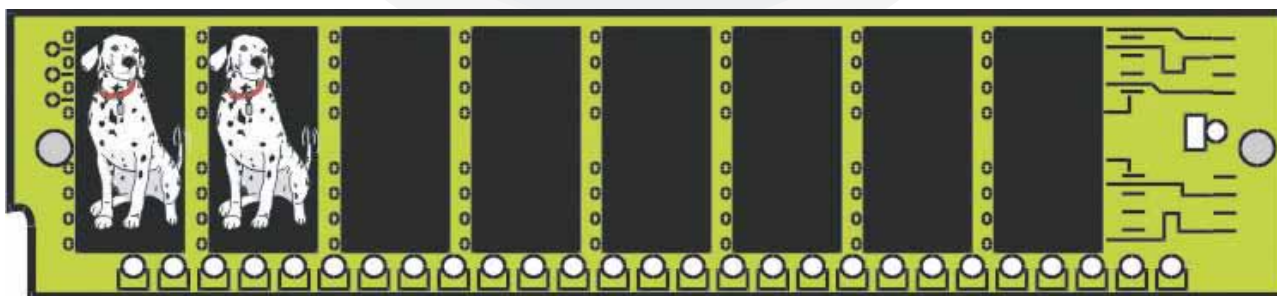


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Um objeto torna-se persistente a partir do momento em que os valores de seus atributos e as informações necessárias para a manutenção de suas conexões com outros objetos são armazenados em uma mídia qualquer.

A persistência refere-se mais a um conceito do que a uma técnica de análise ou de desenvolvimento.

Os objetos criados e destruídos sem que sejam salvos de alguma maneira são chamados **objetos não persistentes**. Esses objetos são temporários, já que os valores de seus atributos são transientes e, por isso, não existe a necessidade ou o interesse de que sejam gravados.

2.6. Abstração

A abstração ocorre quando nos focamos nos aspectos essenciais de uma entidade e ignoramos propriedades secundárias. No desenvolvimento de sistemas, a abstração consiste em nos concentrarmos no que o objeto é e executa, para somente depois tomar decisões sobre sua implementação.

Por meio da abstração, isolamos o objeto que desejamos representar de um ambiente complexo e representamos nele apenas as características relevantes, de acordo com o problema atual.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Assim, ao utilizarmos os conceitos de abstração, as decisões relacionadas ao desenvolvimento e à implementação de objetos podem ser tomadas quando entendemos melhor o problema a ser resolvido. Portanto, a abstração é um dos elementos mais importantes da programação orientada a objetos.

Podemos dizer que a abstração é um processo por meio do qual separamos os fatos relevantes dos detalhes que não têm grande importância. É um conceito bastante adequado em situações de grande complexidade.

Utilizando a abstração, podemos desconsiderar alguns detalhes para ressaltar outros, visto que determinadas partes do problema ganham maior ou menor peso.


A utilização apropriada da abstração permite utilizar em todas as fases de desenvolvimento do sistema (desde a sua análise até a sua documentação) um mesmo modelo conceitual (orientação a objetos).

2.7. Encapsulamento

Diferentemente do que ocorre na programação tradicional, na programação orientada a objetos, no modo de utilização dos atributos e métodos, os dados e as operações realizadas com esses dados são encapsulados em uma única entidade. Assim, a única forma de conhecer ou alterar os atributos de um objeto é por meio de seus métodos. A lista a seguir descreve as vantagens do encapsulamento:

- O objeto é disponibilizado com toda a sua funcionalidade, sem a necessidade de conhecermos seu funcionamento ou armazenamento interno;
- É possível modificar um objeto internamente, acrescentando métodos, sem que isso afete os outros componentes do sistema que utilizam o objeto modificado;
- O processo de desenvolvimento de sistemas é acelerado e simplificado, já que os usuários dos objetos não precisam necessariamente saber como eles são constituídos internamente;
- A implementação de um comportamento pode ser modificada radicalmente sem que haja impacto no resto do programa. Isso é possível porque o código que utiliza o objeto não depende da maneira como ele é implementado.

Para que dois objetos possam interagir, é necessário que um conheça o conjunto de operações disponíveis no outro (interface), e vice-versa, para que possam enviar e receber informações, além de ordenarem a realização de procedimentos.

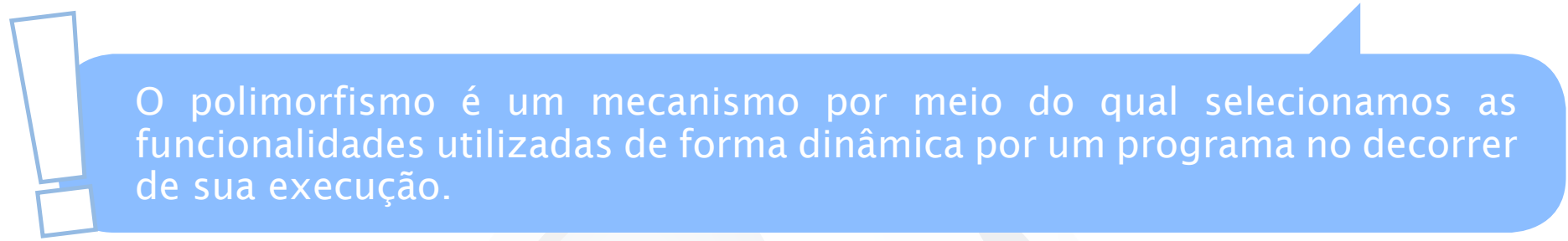


Podemos definir como interface o contrato entre a classe e o mundo exterior. Quando uma classe implementa uma interface, se compromete a fornecer o comportamento publicado por essa interface.

2.8. Polimorfismo

Definimos polimorfismo como um princípio a partir do qual as classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de forma específica para cada uma das classes derivadas.

Com o polimorfismo, os mesmos atributos e objetos podem ser utilizados em objetos distintos, porém, com implementações lógicas diferentes. Por exemplo, podemos assumir que um triângulo e um retângulo são tipos de polígonos e que o cálculo da área de cada um é realizado de forma diferente.



O polimorfismo é um mecanismo por meio do qual selecionamos as funcionalidades utilizadas de forma dinâmica por um programa no decorrer de sua execução.

Para exemplificar o polimorfismo, podemos dizer que uma classe chamada **Cliente** e outra chamada **Funcionario** podem ter como base uma classe chamada **Pessoa**, com um método chamado **Enviar Email**. Se esse método (definido na classe base) se comportar de maneira diferente para as chamadas feitas a partir de uma instância de **Cliente** e para as chamadas feitas a partir de uma instância de **Funcionario**, ele será considerado um método polimórfico, ou seja, um método de várias formas.

2.9. Compartilhamento

Na orientação a objetos, existem técnicas que possibilitam o compartilhamento em vários níveis. Comportamento e herança de estrutura de dados permitem o compartilhamento de estruturas comuns entre classes similares diversas, sem redundância. A utilização da herança no compartilhamento do código traz duas vantagens:

- Economia de código;
- Redução dos casos distintos para análise, já que há uma maior clareza conceitual a partir do reconhecimento de que operações diferentes estão relacionadas ao mesmo elemento.

O desenvolvimento orientado a objetos permite, além do compartilhamento de informação dentro de um projeto, o reaproveitamento de códigos (e até mesmo de projetos) em projetos futuros. A metodologia disponibiliza ferramentas que possibilitam o compartilhamento, como a abstração, a encapsulação e a herança. A reutilização entre projetos pode utilizar como estratégia a criação de bibliotecas de elementos reutilizáveis, além do pensamento do desenvolvedor voltado para termos genéricos, ou seja, não focado apenas na aplicação atual.

The background features a collection of colorful 3D blocks in various colors including purple, blue, yellow, green, orange, and pink. Some blocks are solid, while others are hollow. They are arranged in a scattered, overlapping manner. A large, light blue number '2' is positioned on the right side of the image.

2

Teste seus conhecimentos

Conceitos de orientação a objetos

1. Que conceito define que uma classe pode servir de base para outras classes?

- ☐ a) Polimorfismo
- ☐ b) Herança
- ☐ c) Organização
- ☐ d) Elementos
- ☐ e) Instanciação

2. Como é chamada a classe que serve como modelo para outras classes?

- ☐ a) Classe modelo
- ☐ b) Classe primeira
- ☐ c) Classe derivada
- ☐ d) Classe absoluta
- ☐ e) Classe base

3. Como é chamada a classe que não admite ser instanciada?

- ☐ a) Classe modelo
- ☐ b) Classe abstrata
- ☐ c) Classe oculta
- ☐ d) Classe absoluta
- ☐ e) Classe interna

4. Como é chamado o processo de focar nos aspectos essenciais de uma entidade, deixando a implementação física para depois?

- ☐ a) Análise
- ☐ b) Programação
- ☐ c) Herança
- ☐ d) Abstração
- ☐ e) Modelagem

5. Como é chamado o princípio a partir do qual as classes derivadas de uma única classe são capazes de invocar métodos que se comportam de maneira única?

- ☐ a) Polimorfismo
- ☐ b) Herança
- ☐ c) Abstração
- ☐ d) Encapsulamento
- ☐ e) Nenhuma das alternativas anteriores está correta.

The background of the slide is decorated with a collection of 3D isometric blocks in various colors including purple, blue, yellow, green, orange, and pink. Some blocks are solid, while others are hollow with square cutouts. These blocks are arranged in a scattered, overlapping manner across the slide. Additionally, there are several thin, curved lines in blue, red, and yellow that weave through the blocks, adding a sense of movement and complexity to the design.

Mãos à obra!

Conceitos de orientação a objetos

Laboratório 1

Para este exercício não existe uma única resposta certa e definitiva.

A – Definindo atributos e métodos

1. Defina cinco atributos para cada uma das classes a seguir:

<p>Cliente:</p> <p>1.</p> <p>2.</p> <p>3.</p> <p>4.</p> <p>5.</p>	<p>Produto:</p> <p>1.</p> <p>2.</p> <p>3.</p> <p>4.</p> <p>5.</p>
<p>Boleto:</p> <p>1.</p> <p>2.</p> <p>3.</p> <p>4.</p> <p>5.</p>	<p>Carro:</p> <p>1.</p> <p>2.</p> <p>3.</p> <p>4.</p> <p>5.</p>

2. Defina dois métodos para cada uma das classes a seguir:

<p>Nota fiscal:</p> <p>1.</p> <p>2.</p>	<p>Produto:</p> <p>1.</p> <p>2.</p>
<p>Conta corrente:</p> <p>1.</p> <p>2.</p>	<p>Documento:</p> <p>1.</p> <p>2.</p>

A decorative background featuring a large, stylized number '3' in the upper right corner. The background is filled with numerous 3D blocks of various colors (purple, blue, yellow, green, orange, pink, grey) arranged in a staggered, isometric pattern. Some blocks are solid, while others are hollow. In the lower left, there are some thin, curved lines in blue, purple, and yellow.

3

Notações gráficas de classes e instâncias

- ✓ Definição de OMT;
- ✓ Definição e composição do Modelo de Objetos;
- ✓ Diagrama de classes;
- ✓ Diagrama de instâncias.

3.1. Introdução

Abordaremos, nesta leitura complementar, as notações gráficas do Modelo de Objetos que faz parte da OMT, ou Object Modeling Technique (Técnica de Modelagem de Objetos), desenvolvida por Rumbaugh e utilizada principalmente por desenvolvedores de sistemas e softwares que suportam um ciclo completo de desenvolvimento, objetivando implementações orientadas a objetos.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

A OMT é uma das técnicas de desenvolvimento orientadas a objeto mais populares atualmente e possui uma notação principal simples, o que faz com que seja facilmente compreendida, desenhada e utilizada.

3.2. Modelo de objetos

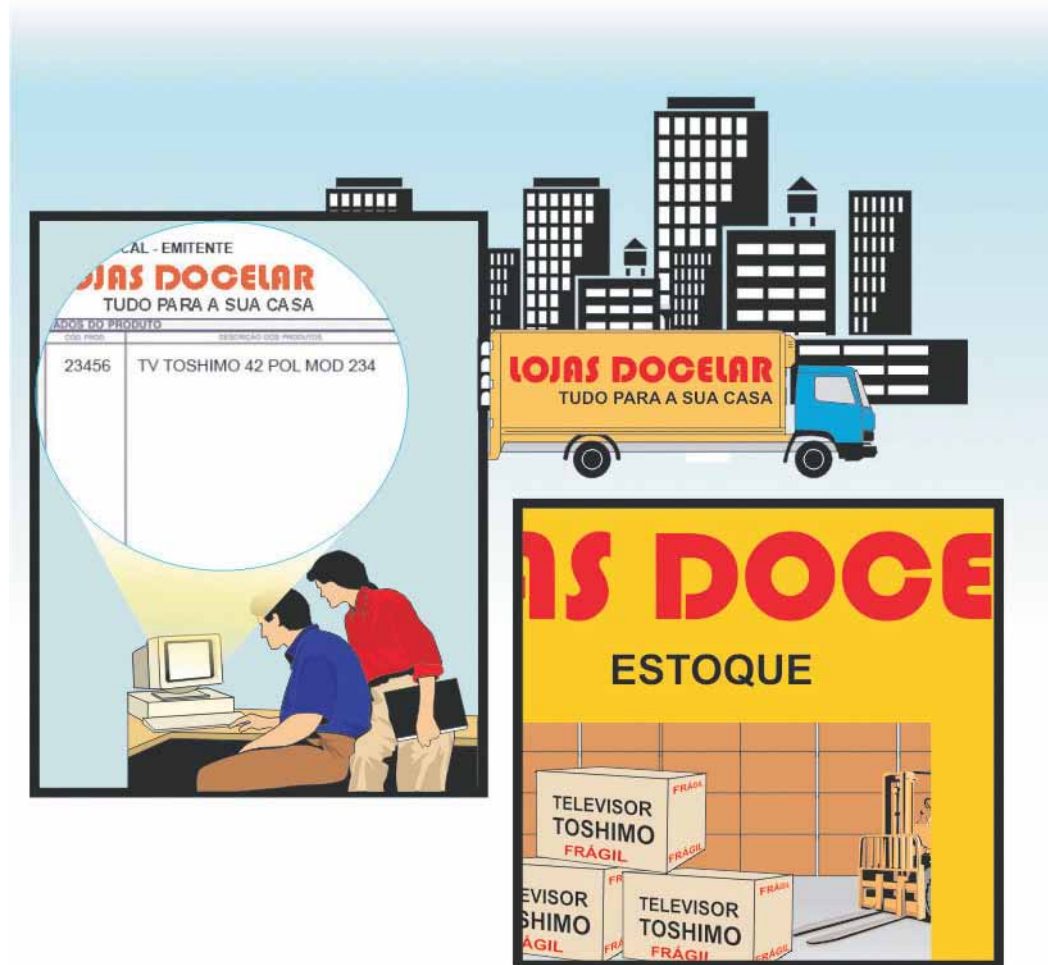


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

A construção do Modelo de Objetos é utilizada para capturar, do mundo real, os conceitos que são importantes para uma aplicação. O Modelo de Objetos descreve a estrutura estática de objetos de um sistema, que inclui:

- A identidade de um objeto;
- Os relacionamentos de um objeto com outros objetos;
- Os atributos de um objeto;
- As operações de um objeto.

O Modelo de Objetos é composto pelo diagrama do modelo de objetos e pelo dicionário de dados, que descreve os atributos (é utilizado para evitar que os atributos sejam explicitados graficamente e para garantir que os diagramas sejam mais administráveis visualmente).

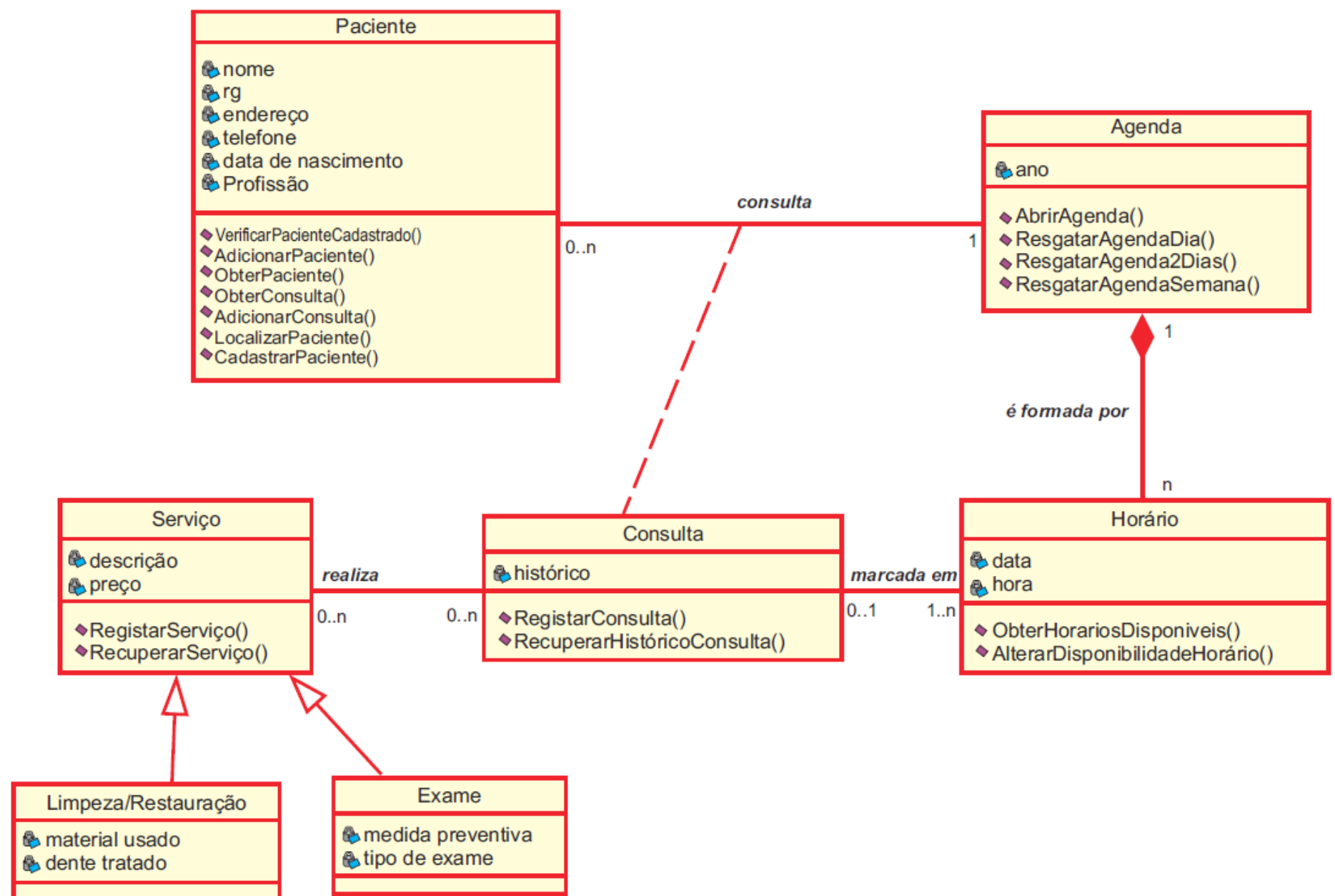


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

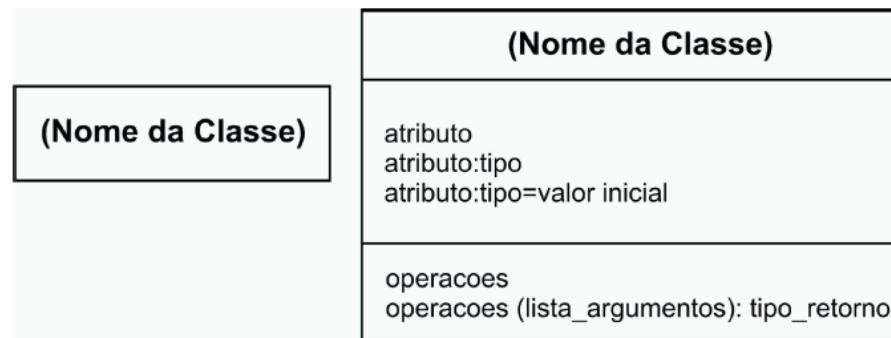
Há dois tipos de diagramas de objetos:

- **Diagrama de classes**: Descreve o caso geral da modelagem;
- **Diagrama de instâncias**: Utilizado para exemplificar.

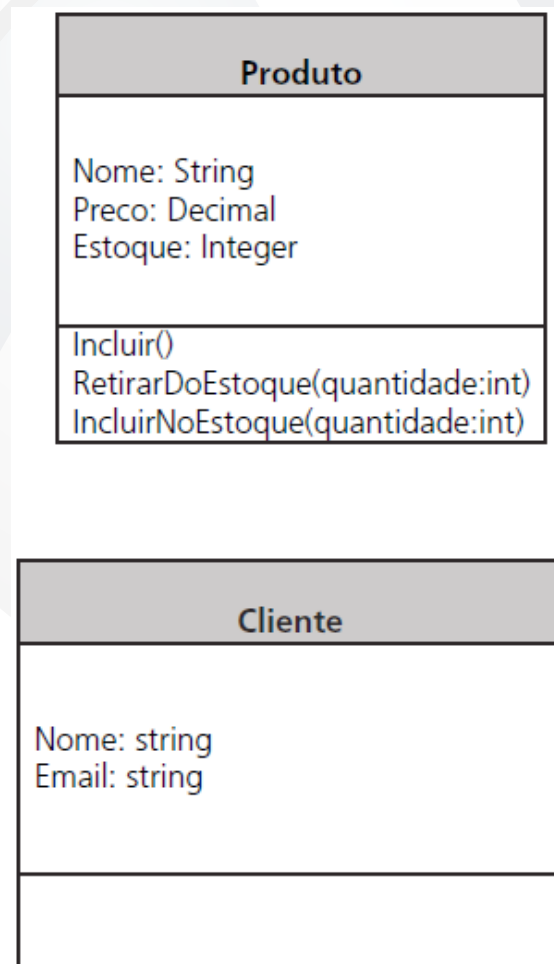
3.2.1. Diagramas de classes

Uma classe é um grupo de objetos que são semelhantes em seus atributos (propriedades), operações (comportamento), relacionamentos com outros objetos e semântica. Cada classe possui um grupo de atributos e operações.

A notação OMT para uma classe é um retângulo contendo o nome da classe em negrito e seções opcionais para os atributos e operações (essas seções devem ser separadas por linhas horizontais):



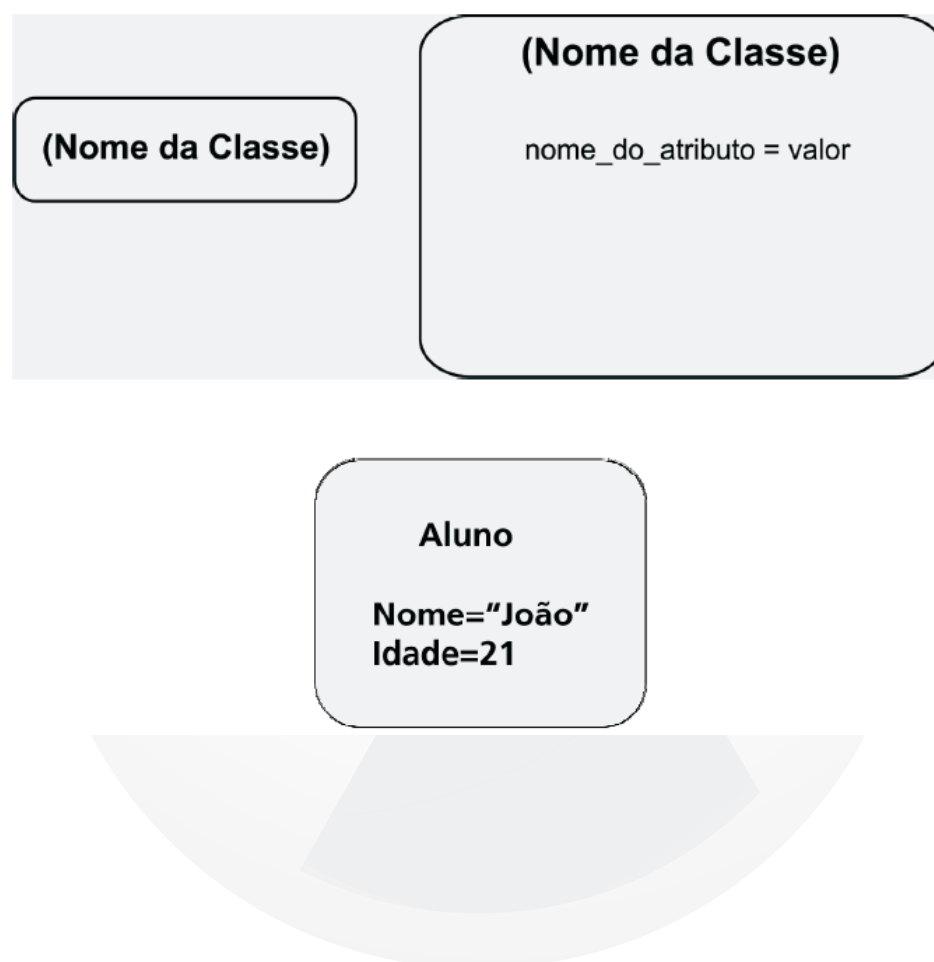
Exemplos:



3.2.2. Diagramas de instâncias

Utilizados na documentação de testes e na apresentação de resultados, os diagramas de instâncias, também conhecidos como diagramas de objetos, descrevem os relacionamentos de um grupo particular de objetos.

A notação OMT para instâncias é um retângulo de cantos arredondados, que inclui o nome da classe em negrito e entre parênteses. Opcionalmente, pode incluir os valores dos atributos separados do nome da classe por uma linha horizontal:



The background features a collection of colorful 3D blocks in various colors (purple, blue, yellow, green, orange, pink, grey) arranged in a scattered, overlapping manner. A large, light blue number '3' is positioned on the right side of the image.

3

Teste seus conhecimentos

Notações gráficas de classes e instâncias

1. Qual elemento não faz parte da estrutura estática de um objeto?

- ☐ a) Identidade
- ☐ b) Atributos
- ☐ c) Relacionamentos
- ☐ d) Compatibilidade com sistema operacional
- ☐ e) Operações

2. Como é chamado o componente que descreve os atributos de um objeto?

- ☐ a) Classe modelo
- ☐ b) Dicionário de dados
- ☐ c) Objeto derivado
- ☐ d) Herança múltipla
- ☐ e) Gerenciador de atributos

3. O que é o atributo de um objeto?

- ☐ a) Uma classe.
- ☐ b) Uma instância.
- ☐ c) Uma propriedade.
- ☐ d) Uma operação.
- ☐ e) Nenhuma das alternativas anteriores está correta.

4. Uma operação do objeto está associada a qual característica?

- ☐ a) Comportamento
- ☐ b) Herança
- ☐ c) Polimorfismo
- ☐ d) Abstração
- ☐ e) Modelagem

5. O que representa a parte inferior de um diagrama de classe?

- ☐ a) O nome.
- ☐ b) As propriedades.
- ☐ c) Os métodos.
- ☐ d) O relacionamento.
- ☐ e) Nenhuma das alternativas anteriores está correta.

The background features a collection of 3D cubes in various colors (purple, blue, yellow, green, orange, pink, grey) arranged in a scattered, overlapping pattern. Some cubes are solid, while others have a square hole in the center. Thin, curved lines in blue, red, and green pass through the scene, adding a sense of movement and depth.

Mãos à obra!

Notações gráficas de classes e instâncias

Laboratório 1

Para este exercício não existe uma única resposta certa e definitiva. Modelos diferentes de dados podem atingir o mesmo objetivo. O importante é treinar a capacidade de abstração.

A – Criando classe e desenhando diagrama

1. Crie a classe e desenhe o diagrama a partir do enunciado a seguir. Não é necessário criar os métodos com os relatórios, apenas a classe de dados com informações suficientes para desenvolver o programa.

A empresa X é uma clínica médica que precisa automatizar o processo de marcação de consulta de seus pacientes. O processo de atendimento é o seguinte:

- O paciente liga solicitando uma consulta com um médico de uma determinada especialidade;
- A atendente procura na agenda um horário livre de um médico daquela especialidade;
- A atendente marca a consulta;
- No dia da consulta, a atendente recebe o valor da consulta se o paciente não estiver inscrito em um dos convênios médicos com os quais a clínica trabalha;
- Se a consulta do paciente for um retorno, não é cobrada;
- O valor da consulta varia de acordo com a especialidade;
- Cada médico tem apenas uma especialidade;
- O médico pode solicitar a qualquer momento a sua agenda do dia, da semana, do mês ou de qualquer período à atendente;
- O paciente pode ligar perguntando quando é a sua próxima consulta;

- O contador da clínica pode ligar solicitando uma lista de tudo que foi pago e tudo o que deve ser recebido dos planos de saúde;
- Cada plano de saúde deve receber uma lista dos atendimentos efetuados no mês para proceder ao pagamento para a clínica;
- O médico pode solicitar um histórico do paciente, ou seja, todas as vezes que ele veio a uma consulta e com qual médico se consultou;
- Cada médico trabalha exclusivamente na clínica, ou seja, está disponível em todos os horários;
- As consultas são de hora em hora.



4

Estruturas e relacionamentos

- ✓ Generalização e herança;
- ✓ Agregação;
- ✓ Ligações e associações;
- ✓ Compartilhamento.

4.1. Introdução

Nesta leitura complementar, apresentaremos uma parte do Modelo de Objetos (um subconjunto do modelo OMT), que retrata o relacionamento entre objetos. Esse modelo pode ser introduzido a partir do estabelecimento dos principais conceitos e definições da abordagem de orientação a objetos.

4.2. Generalização e herança

A generalização e a herança são abstrações que permitem que classes compartilhem similaridades ao mesmo tempo em que preservam as características que as diferem.

O relacionamento de uma classe com suas versões refinadas, ou seja, especializadas, é chamado de generalização. Nesse ponto, é importante definir dois conceitos:

- **Superclasse ou classe base:** É a classe por meio da qual podemos gerar subclasses; também é conhecida como classe pai;
- **Subclasse ou classe derivada:** É a versão refinada de uma superclasse.

Vejamos um exemplo: pode haver uma classe base chamada **Pessoa**, com características comuns a todas as pessoas: **Nome**, **Email**, **Telefone**. Uma classe derivada poderia ser a classe **Funcionario**, que, além de ter os campos **Nome**, **Email** e **Telefone**, teria também os campos **Salário** e **Cargo**. Uma outra classe derivada poderia ser a **Cliente**, que, além dos campos herdados da classe base, teria o campo **Nome do Contato** ou **Celular**.



As classes base melhoram a consistência de um sistema, facilitando a padronização dos métodos e propriedades. Os principais ambientes de programação do mercado, como o .NET Framework da Microsoft ou o Java da Sun Systems, foram construídos usando o sistema de classes base e classes derivadas.

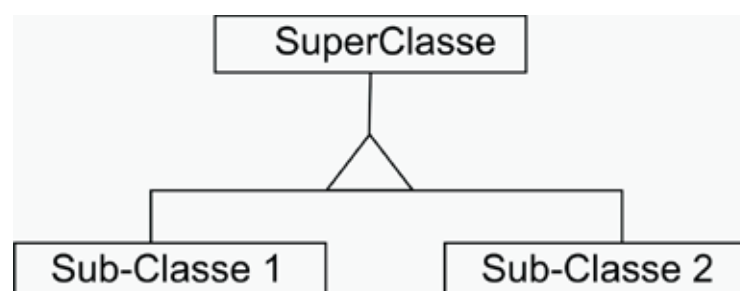
A generalização também pode ser chamada de relacionamento **is-a** (ou seja, **é-um**), já que toda instância de classe derivada também é uma instância de classe base. As características da classe base são herdadas pela classe derivada, e as operações e os atributos comuns a um conjunto de classes derivadas são colocados como atributos e operações da classe base.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

A generalização é a estrutura que permite a utilização do conceito de herança, sendo aplicada a diversos níveis. Além das características herdadas de seus ancestrais, uma classe derivada pode acrescentar operações e atributos específicos.

A imagem seguinte exibe a notação diagramática de OMT utilizada para representar a generalização: um triângulo cujo vértice aponta para a classe base.

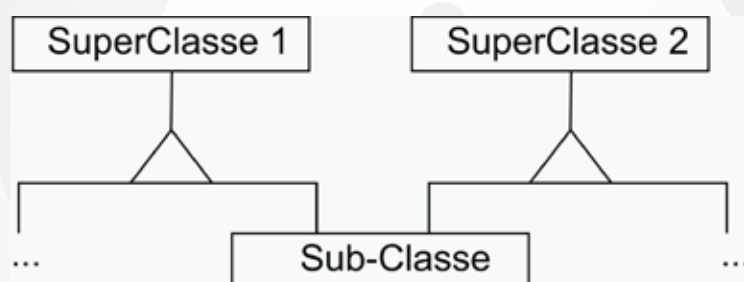


Cada associação do tipo generalização pode possuir um discriminador associado. O discriminador é um atributo (do tipo enumeração) utilizado para indicar qual é a propriedade do objeto que é abstraída pelo relacionamento de generalização. Trata-se de um nome para a base de generalização.

Uma característica de classe base pode ser sobreposta por uma classe derivada, caso esta defina uma característica própria com o mesmo nome. A característica própria da classe derivada refinará e substituirá a característica da classe base.

As características podem ser sobrepostas por questões de desempenho ou de refinamento de especificação, por exemplo. As características que podem ser sobrepostas são os valores default dos atributos e métodos de operação, entre outras. É importante lembrar que uma característica não deve ser sobreposta de forma inconsistente com a semântica da classe base.

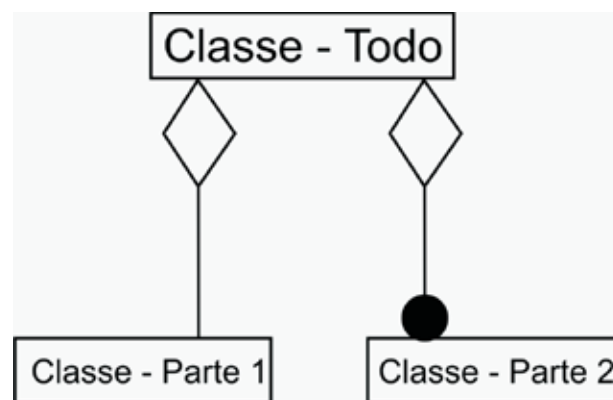
Temos a herança múltipla representada graficamente na imagem a seguir:



A herança múltipla não é aceita em todos os ambientes de programação orientados a objetos.

4.3. Agregação

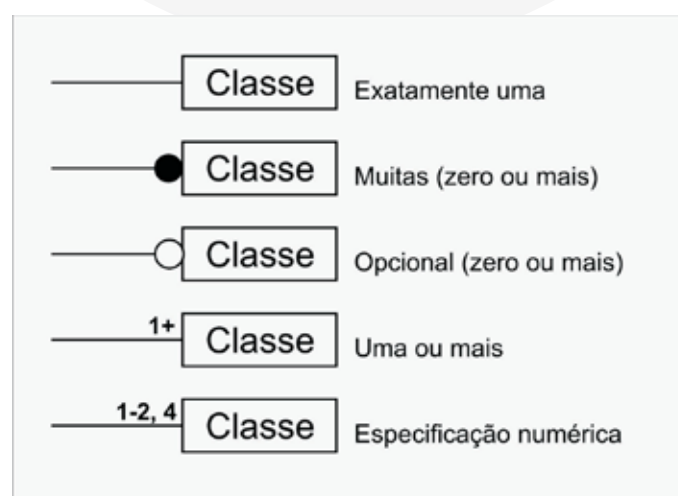
Na agregação, há certa coesão entre as partes, porém, elas não são totalmente dependentes. Para exemplificar, podemos imaginar a criação de uma nova classe, **Família**, constituída de vários membros da classe **Pessoa**, porém, os membros da classe **Pessoa** podem existir fora da classe **Família**.



A agregação define um relacionamento do tipo **uma-parte-de**. Nesse tipo de relacionamento, alguns objetos (parte) representam componentes de outro objeto (todo), porém, um não contém o outro, o todo não contém a parte, não havendo relação de posse exclusiva. Em uma agregação, um componente que faz parte de outro pode existir isoladamente.

Isso é diferente do que acontece na composição, como em uma nota fiscal, por exemplo, em que o objeto que representa um produto comprado pertence a essa nota e apenas a ela, não fazendo sentido esse objeto existir se não for dentro dessa nota fiscal.

As representações gráficas para tipos diferentes de agregação são exibidas na próxima imagem. Essas notações também podem ser utilizadas para associações e para ligações.



4.3.1. Conexões entre objetos

Conexões de ocorrência e conexões de mensagens são tipos de conexão entre objetos que não caracterizam um tipo de hierarquia ou de estrutura.

4.3.1.1. Conexão de ocorrência

Este tipo de conexão ocorre quando um atributo de um objeto faz referência a outro objeto. As conexões de ocorrência normalmente são criadas quando identificamos que atributos redundantes de um objeto são parte de outro objeto.

As conexões de ocorrência possibilitam também o registro do número de vezes que um objeto faz referência a outro ou é referenciado, o que também é chamado de cardinalidade.

4.3.1.2. Conexão de mensagem



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Este tipo de conexão ocorre quando um objeto envia uma mensagem para outro objeto. No contexto de uma conexão de mensagem, existem os seguintes elementos:

- **Objeto transmissor:** O objeto que transmite a mensagem para um outro objeto;
- **Mensagem:** Procedimento que dispara um método específico para que o objeto receptor execute um comportamento determinado. Pode ser uma solicitação de informações ou uma solicitação para a realização de alguma ação no objeto;
- **Objeto receptor:** O objeto que recebe a mensagem e retorna ou não uma resposta para o objeto transmissor.

4.4. Ligações e associações

Ligações e associações são mecanismos que estabelecem relacionamentos entre objetos e classes. A seguir, cada um desses mecanismos será descrito:

- **Ligações**

Uma ligação conecta duas instâncias de objetos, de forma física ou conceitual, por exemplo, **Paulo** é aluno da **Faculdade_Impacta**. Uma ligação é uma instância de uma associação.

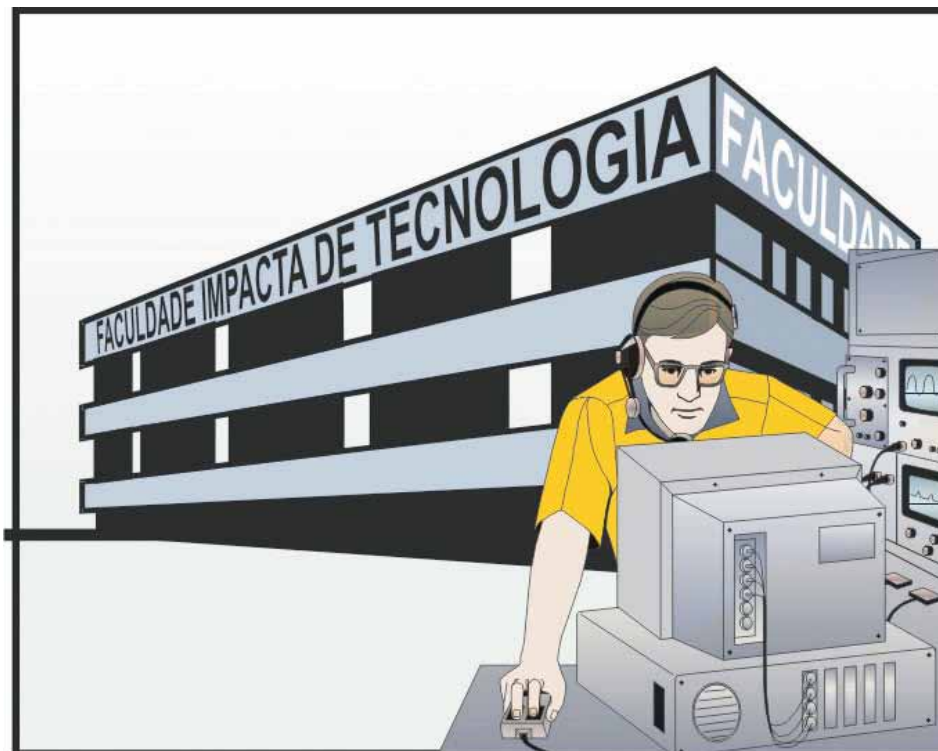
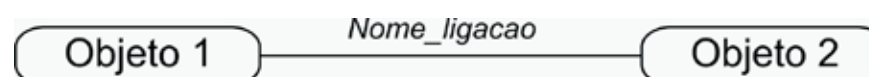


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Na notação de diagramas OMT, uma ligação é representada como uma linha que conecta dois objetos. A imagem seguinte exemplifica um diagrama OMT com ligação:



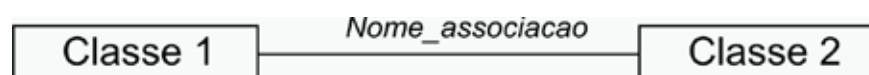
- **Associações**

As associações definem um conjunto de ligações que compartilham a mesma semântica e estrutura, por exemplo, **uma pessoa é aluna de uma faculdade**. Podemos entender que, assim como as classes descrevem grupos de objetos potenciais, as associações descrevem grupos de ligações potenciais.

Introdução à Programação Orientada a Objeto (online)

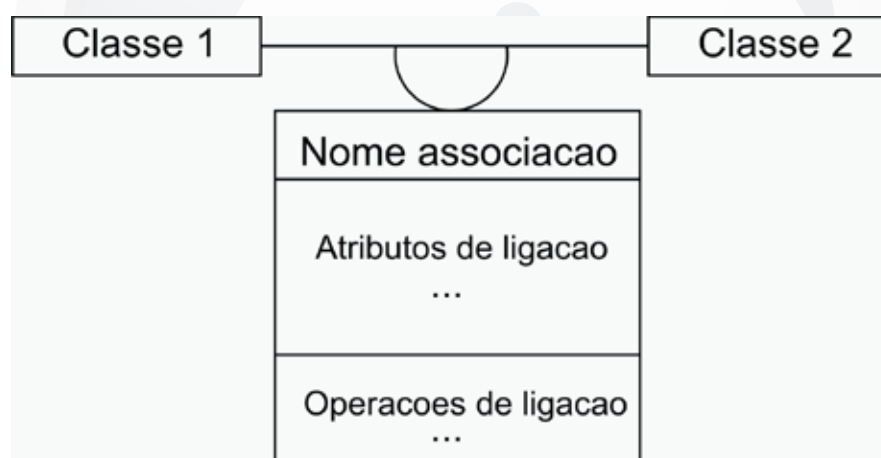
66

Uma associação é representada por uma linha que conecta duas classes. A próxima imagem exemplifica um diagrama OMT com associação:



Na notação de diagramas **OMT**, normalmente, os nomes das associações e das ligações são exibidos em estilo itálico e podem ser omitidos quando só existir uma associação de sentido óbvio entre um par de classes, para as associações, ou entre um par de objetos, para as ligações.

A OMT introduz o conceito de atributo de ligação para os casos em que os atributos dizem respeito a associações e não a classes. A associação pode ser modelada como uma classe conectada à associação quando possuir operações associadas:



The background features a collection of 3D cubes in various colors (purple, blue, yellow, green, orange, pink, grey) arranged in a scattered, overlapping pattern. A large, bold blue number '4' is positioned on the right side of the image.

4

Teste seus conhecimentos
**Estruturas e
relacionamentos**

1. Como é chamado o relacionamento de uma classe com suas versões especializadas?

- ☐ a) Generalização
- ☐ b) Protótipo
- ☐ c) Usuário
- ☐ d) Objeto
- ☐ e) Identidade

2. Que outra denominação é dada à classe base?

- ☐ a) Classe Master
- ☐ b) Classe Mestre
- ☐ c) Classe Única
- ☐ d) Classe Derivada
- ☐ e) Classe Pai

3. Considere uma classe que representa uma nota fiscal de venda. Um produto comprado, descrito nesta nota fiscal, não existe isoladamente. Como é chamada a propriedade de um objeto existir apenas dentro de outro?

- ☐ a) Instância
- ☐ b) Propriedade
- ☐ c) Operação
- ☐ d) Agregação
- ☐ e) Relacionamento

4. Como é chamada a conexão resultante quando um atributo de um objeto faz referência a outro objeto?

- ☐ a) Conexão de objeto
- ☐ b) Conexão de mensagem
- ☐ c) Conexão de ocorrência
- ☐ d) Conexão simples
- ☐ e) Conexão complexa

5. Como é chamado um conjunto de ligações com características comuns?

- ☐ a) Associação
- ☐ b) Propriedade
- ☐ c) Métodos
- ☐ d) Eventos
- ☐ e) Nenhuma das alternativas anteriores está correta.

The background of the entire page is an abstract composition of 3D cubes in various colors (purple, blue, yellow, green, orange, pink, grey) arranged in a staggered, overlapping pattern. Some cubes are solid, while others are hollow. Thin, curved lines in blue, red, and yellow weave through the cubes, adding a sense of movement and connection. The overall aesthetic is clean, modern, and geometric.

Mãos à obra!

Estruturas e relacionamentos

Laboratório 1

Para este exercício não existe uma única resposta certa e definitiva. Modelos diferentes de dados podem atingir o mesmo objetivo. O importante é treinar a capacidade de abstração.

A – Criando classe e desenhando diagrama

1. Crie a classe e desenhe o diagrama a partir do seguinte enunciado:

A empresa X é uma loja de departamentos e precisa abrir uma loja on-line. A loja terá produtos em promoção na página inicial e um menu apontando para as categorias de produtos. O cliente poderá clicar na categoria de produto para acessar a parte do site daquela categoria. Se ele clicar em um produto, os dados deverão entrar no seu carrinho de compras. Ao fechar a venda, um cadastro será feito com os dados do cliente e um boleto será emitido (única forma de pagamento nesta primeira etapa). O produto será retirado do estoque e entregue à transportadora. O cliente deverá poder acompanhar o processamento do seu pedido e de todas as compras feitas na loja.

Além do site, é necessária uma intranet para que os funcionários da empresa cadastrem novos produtos e alterem os preços, além de poderem cadastrar os pedidos feitos aos fornecedores. Funcionários, Clientes e Fornecedores devem conter em seu cadastro Nome, E-mail, Telefone, Cidade, Estado e CEP.

The background features a large, semi-transparent number '5' on the right side. Scattered across the page are numerous 3D cubes in various colors including purple, blue, yellow, orange, green, and grey. Some cubes are solid, while others are hollow. Thin, curved lines in blue, green, and yellow weave through the cubes, adding a sense of movement and connectivity.

Ambientes de desenvolvimento de software

- ✓ O que é um software;
- ✓ Tipos de software;
- ✓ Linguagens de programação;
- ✓ Bancos de dados;
- ✓ Tecnologias e ferramentas;
- ✓ Frameworks;
- ✓ Metodologias.

5.1. Introdução

Nesta leitura complementar, veremos como a Programação Orientada a Objetos se apresenta nos ambientes de desenvolvimento de software e quais as tecnologias e profissionais envolvidos nesse processo.

5.2. O que é um software

Software é um conjunto de instruções a serem executadas por um computador com o objetivo de resolver algum problema. Desenvolver um software é elaborar tais instruções de maneira lógica e ordenada, em um formato que o computador consiga interpretar.

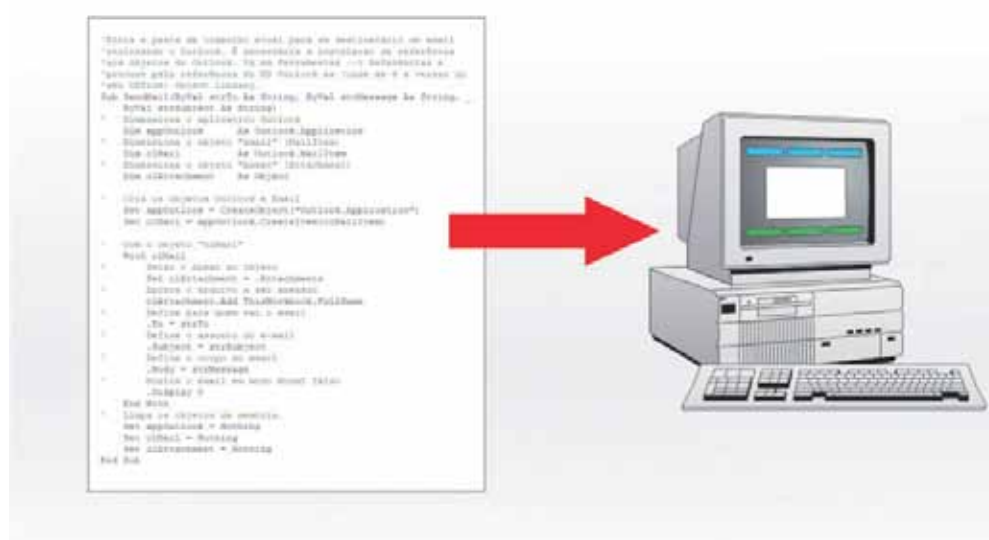


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

A elaboração de um software passa por cinco fases: **Conceito**, **Design**, **Desenvolvimento**, **Testes** e **Implantação**. A seguir veremos o que acontece em cada uma dessas fases.

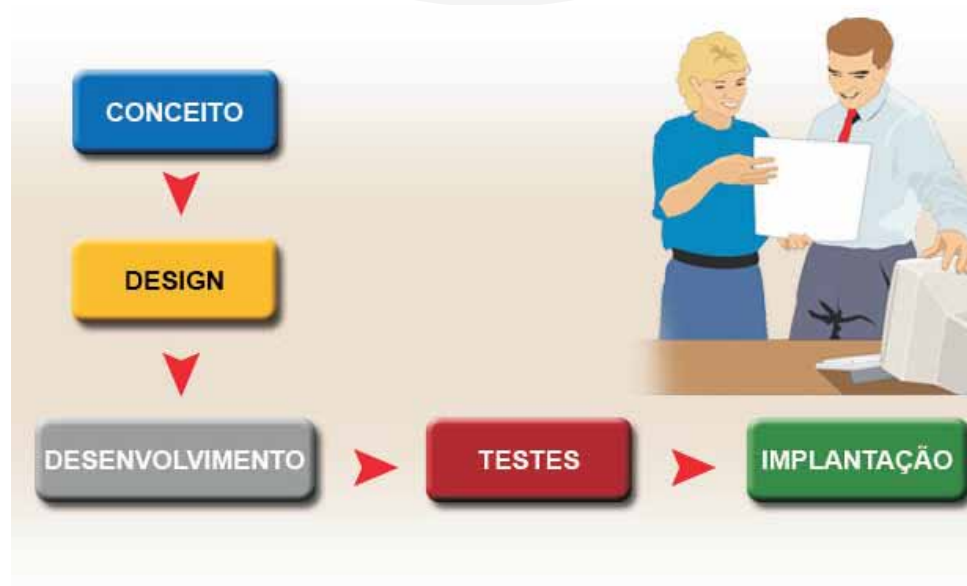


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

- **Fase 1 – Conceito**

Nesta fase, o problema a ser resolvido é entendido pelos analistas para que seja possível construir a solução da melhor maneira possível. Esta é a fase mais importante do projeto, porque servirá de guia para todo o restante do processamento.

Normalmente, diversas reuniões são agendadas com os analistas para que o cliente explique detalhadamente a sua necessidade. O cliente é quem solicita o software, ou seja, é a pessoa ou empresa que tem uma necessidade específica.

- **Fase 2 – Design**

Nesta fase serão escolhidas as tecnologias que melhor atenderão às necessidades dos clientes, assim como a equipe necessária, o prazo e o custo do projeto. Normalmente, esta etapa é feita com os analistas trabalhando em conjunto com os programadores e técnicos. O resultado final desta fase são diversos diagramas em UML ou outra linguagem de especificação.

- **Fase 3 – Desenvolvimento**

Nesta fase entra o trabalho dos programadores, técnicos, especialistas em bancos de dados (dbas) e designers. É a construção do software, feita de acordo com as especificações definidas na fase 2. Normalmente, o processo é feito em etapas, com cronogramas e datas de entrega de partes funcionais do software.

- **Fase 4 – Testes**

A fase de testes consiste em verificar se as regras definidas na fase 2 atendem às necessidades definidas na fase 1. Muitas vezes, neste ponto o desenvolvimento retorna à fase 2 para redesenhar parte do projeto ou, às vezes, para a fase 1, para um melhor entendimento das regras do negócio.

Outra parte importante desta etapa é a estabilização do software. Cada funcionalidade é testada e pequenos ajustes são feitos até que fique totalmente funcional e livre de erros.

- **Fase 5 – Implantação**

Uma vez testado, o software pode ser implantado no ambiente de produção. Muitas vezes é necessário outro projeto apenas para esta fase, principalmente quando já existe um sistema e os dados necessitam de migração. Às vezes é necessário voltar às fases anteriores para ajustar algo que funcionou bem no ambiente de testes, mas que se mostrou ineficiente no ambiente de produção.

5.2.1. Tipos de software

Um software é composto por diversos tipos de projetos. Às vezes, um único projeto é suficiente e, às vezes, são necessários mais de 100 projetos para criar uma solução que atenda às necessidades de uma empresa.

5.2.1.1. Interface de usuário (User Interface ou UI)

A UI é a parte do software com a qual o usuário final interage. Quando uma pessoa utiliza um caixa eletrônico de um banco, por exemplo, os botões na tela e as mensagens são interfaces de usuário. Quando o usuário confirma uma transferência de saldo, não é a interface que faz a transação, mas outro componente com o qual a interface se comunica.

Existem diversas interfaces:

- Programas Windows, como Excel ou Word;
- Programas Web, como lojas on-line;
- Programas para celular;
- Programas para dispositivos portáteis;
- Programa para o surface da Microsoft.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

5.2.1.2. Componentes

Os componentes são a parte do software que faz algum processamento, mas que não interage com o usuário final diretamente. Por exemplo, ao confirmar uma transação em um caixa eletrônico, um componente deve guardar essa informação em um banco de dados. Mas esse componente pode ser, também, acionado pela Internet ou por um celular. Esse componente não está ligado a uma interface em particular, ele apenas recebe uma mensagem e realiza a tarefa de gravação.

Existem diversos tipos de componentes:

- Componentes para gravar em bancos de dados;
- Componentes distribuídos, para serem chamados de outro computador;
- Componentes contendo grupos de controles para criar interfaces de usuários.

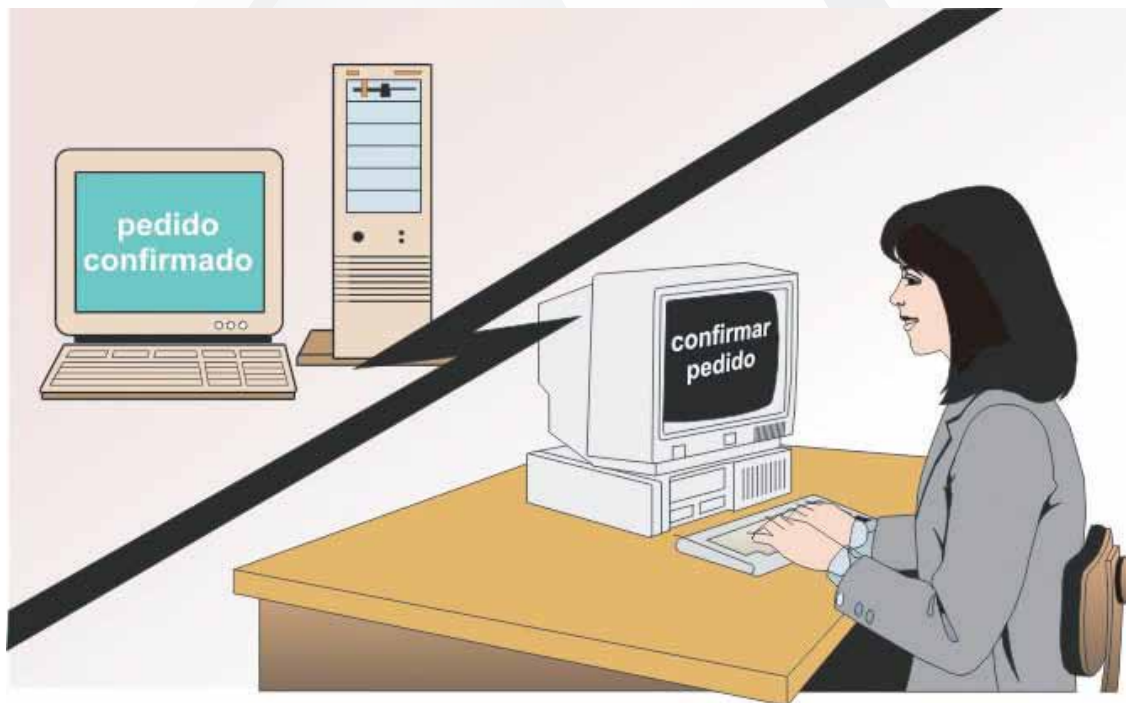


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

5.2.1.3. Serviços

O Windows utiliza o conceito de serviços, que é um software ativado automaticamente quando o computador é ligado. Esse tipo de software é chamado de servidor. Por exemplo, os softwares antivírus são geralmente instalados como serviços no computador. Assim, o usuário não precisa se lembrar de chamar o programa toda vez que ligar o computador e, além disso, muitos vírus entram em ação antes de o usuário fazer o login na máquina.

5.2.1.4. Web services

Os Web services são componentes instalados em um servidor Web. A vantagem de um Web service é fazer com que um computador que não está na rede local possa chamar serviços de outro computador pela Internet. Em uma empresa, em que normalmente os firewalls bloqueiam qualquer acesso, exceto acesso à Internet, isso é claramente uma vantagem.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

5.3. Linguagens de programação

Uma das decisões que um analista deve tomar é a tecnologia que será utilizada para construir um software. Isso envolve escolher o sistema operacional, o ambiente de execução do software, a linguagem de programação, as ferramentas de desenvolvimento, o banco de dados, a metodologia de desenvolvimento e os profissionais necessários.

Há alguns anos, a única preocupação de um analista era a linguagem de programação e todo o resto dependia dessa escolha. A partir do ano 2000, essa abordagem foi mudando, pois cada vez mais uma linguagem pode utilizar componentes feitos por outra e até desenvolvidos em outro sistema operacional. A linguagem de programação passou a ser um dos muitos fatores a serem considerados.

Vejamos algumas linguagens existentes:

Clipper	T-SQL
COBOL	Ruby
Visual Basic	ActionScript
Java	Assembly
C++	Pascal
C#	Python
Visual Basic .NET	PHP
JavaScript	Self
VBScript	OCaml
J#	Fortran

Não existe uma resposta simples para a pergunta: “Qual linguagem escolher para iniciar um software?”. Qualquer linguagem pode ser usada para fazer um bom software, assim como qualquer linguagem pode ser usada para desenvolver um péssimo software. Não é a linguagem que determina o sucesso, mas a mente do profissional que concebeu e desenvolveu o software.

Vejamos algumas considerações sobre as linguagens de programação:

- A linguagem Java é muito difundida e conta com uma comunidade muito grande de programadores e colaboradores. Ela é gratuita, portanto, não há custo com ferramenta de desenvolvimento. O profissional da linguagem Java é um dos mais bem pagos do mercado e existe uma procura muito grande por bons programadores;
- A linguagem C# (C Sharp) é uma iniciativa da Microsoft e foi construída especialmente para a plataforma .NET, que é uma coleção de programas e ferramentas para desenvolver e executar softwares usando a Internet como ambiente. Existe uma grande procura por profissionais desta linguagem no mercado. A Microsoft disponibilizou ferramentas gratuitas para criar programas com esta linguagem, mas a principal ferramenta, o Visual Studio, é um programa pago;
- A linguagem PHP é muito utilizada para criar páginas Web, é de fácil aprendizado e largamente utilizada. As últimas versões incorporaram recursos de orientação a objetos. O mercado para PHP é menor do que para Java e C#, mas existe muita procura por profissionais, principalmente para dar suporte e desenvolver planos de migração para as linguagens mais atuais;
- A linguagem VBScript, utilizada no ambiente de programação para Web Active Server Pages (ASP), foi substituída pela linguagem Visual Basic .NET e pelo ambiente ASP.NET, mas ainda é utilizada em grande parte dos sites. A procura por profissionais é essencialmente para migração de dados;

- Ruby é uma linguagem desenvolvida especialmente para a Web e largamente difundida fora do Brasil. Existem diversas bibliotecas para esta linguagem, sendo mais famosa a chamada Ruby on Rails. Uma das características é a criação rápida de sites.

5.4. Bancos de dados

Uma parte fundamental de uma aplicação é como as informações são gravadas e como são recuperadas. A escolha de um banco de dados é fundamental para o sucesso do software.

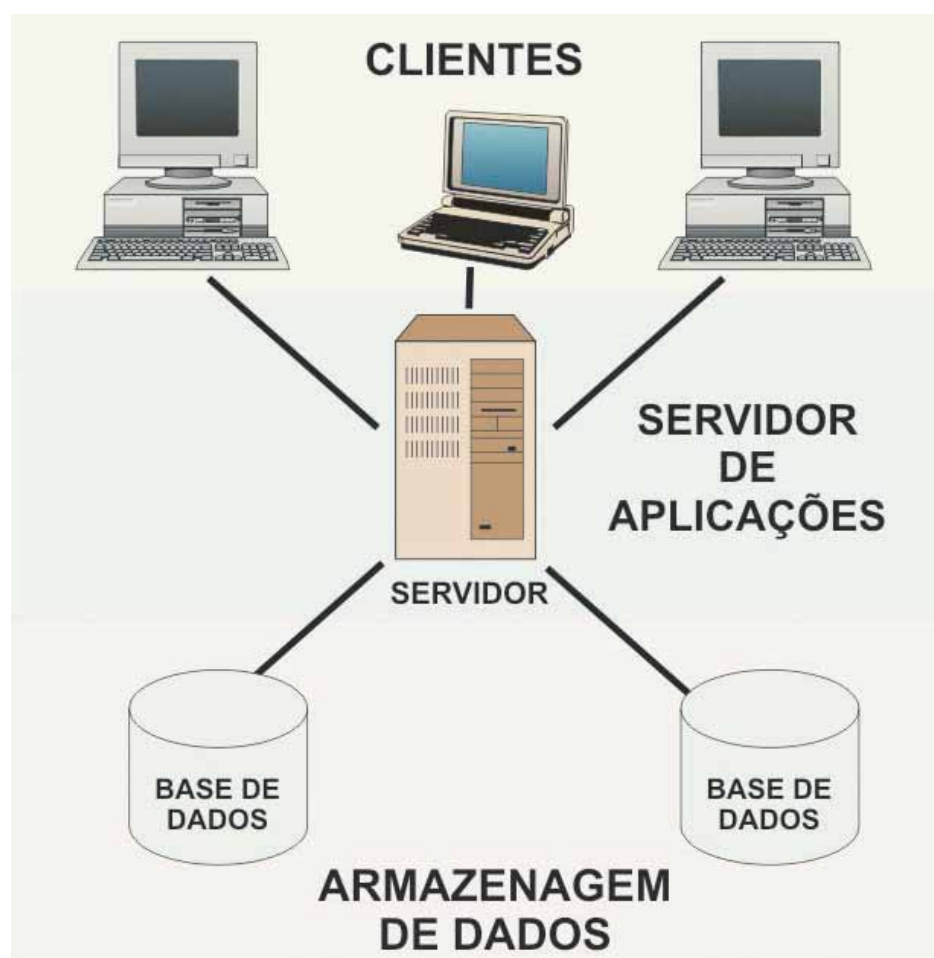


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

Existem dois tipos de bancos de dados: de servidor e de arquivo.

- **Banco de dados de arquivo**

Neste tipo de banco de dados, as informações são gravadas diretamente em um ou mais arquivos, e estes são abertos diretamente pelo sistema operacional. Este tipo de banco é ideal para aplicações isoladas, que não utilizam a rede. Ao compartilhar o arquivo em rede, corre-se o risco de corromper o arquivo caso haja uma falta de energia no momento de uma gravação ou mesmo se houver um problema com a placa de rede.

Podemos citar como exemplos de bancos de dados de arquivo: Access, DBF, Excel, Paradox.

- **Banco de dados de servidor**

Neste tipo de banco de dados, um serviço instalado em um computador (chamado servidor) acessa os dados e cuida da integridade dos mesmos. Os aplicativos e usuários acessam o servidor e não os arquivos diretamente. Este tipo de banco de dados é ideal para aplicações multiusuários e para grande volume de dados.

Podemos citar como exemplos de banco de dados de servidor: SQL Server, Oracle, MySQL, PostgreSQL, DB2, Firebird.

A linguagem padrão para o acesso a um banco de dados é a SQL (Structured Query Language). Cada banco pode introduzir variações e funções específicas, mas o padrão é praticamente o mesmo para todos os bancos.

5.5. Tecnologias e ferramentas

Existem muitas ferramentas disponíveis para ajudar um profissional a desenvolver qualquer parte de um software. As ferramentas que reúnem em um único lugar todos os recursos necessários para criar um software são chamadas IDE ou Ambiente de Desenvolvimento Integrado.

5.5.1. Java

O ambiente Java conta com aplicativos que facilitam o design de aplicações. As ferramentas mais conhecidas são:

- NetBeans;
- Eclipse;
- jEdit;
- JBuilder;
- JDeveloper.

5.5.2. Plataforma .NET

A plataforma .NET (Linguagens VB.NET e C#) conta com muitas ferramentas para criar softwares desde a concepção até a implantação final. Algumas dessas ferramentas são:

- Visual Studio;
- Web Developer;
- C# Express;
- VB Express.

5.6. Frameworks

Frameworks são conjuntos de códigos compilados prontos para serem usados em um aplicativo. O uso de um framework pode acelerar o processo de desenvolvimento de um software. A decisão de usar ou não um framework é geralmente polêmica dentro de um ambiente corporativo. De um lado, o framework pode tornar mais ágil o desenvolvimento, mas por outro pode limitar seriamente a construção de algo novo.

Alguns frameworks famosos são:

- **Hibernate**

O Hibernate é um framework que pode ser usado tanto em Java quanto em .NET (C#/VB). O objetivo é transformar os objetos em instruções SQL para automatizar o processo de gravação e leitura de objetos. Esse tipo de operação é chamada de Mapeamento Objeto-Relacional.

- **AJAX.NET**

Ajax.NET é uma biblioteca para utilizar Ajax em aplicações Web. O Ajax é uma tecnologia que combina JavaScript com XML para fazer as páginas Web responderem mais rapidamente.

- **Struts**

Struts é um framework para realizar o controle de operações com objetos com validação, gravação e recuperação de dados. O Struts foi desenvolvido em Java.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

5.7. Metodologias de desenvolvimento

Construção de software é uma ciência recente e em fase de estruturação. Várias metodologias de desenvolvimento têm aparecido e desaparecido com o passar dos anos. Na verdade, cada empresa acaba criando uma metodologia própria de desenvolvimento com o passar do tempo, baseada em suas necessidades, disponibilidade de mão de obra, prazos, custos e casos de sucesso e fracasso.

Algumas metodologias famosas são:

- **Programação estruturada**

É uma forma de programa em sequência, priorizando as ações a serem tomadas. Muito popular nos anos 1960, foi sucedida pela Programação orientada a objetos.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

- **RUP**

O Rational Unified Process é uma metodologia orientada a objetos cujo ponto principal é a notação UML para definir os procedimentos. É geralmente usado para grandes projetos, quando existe um analista ou uma equipe de analistas que determina minuciosamente o que deve ser feito.

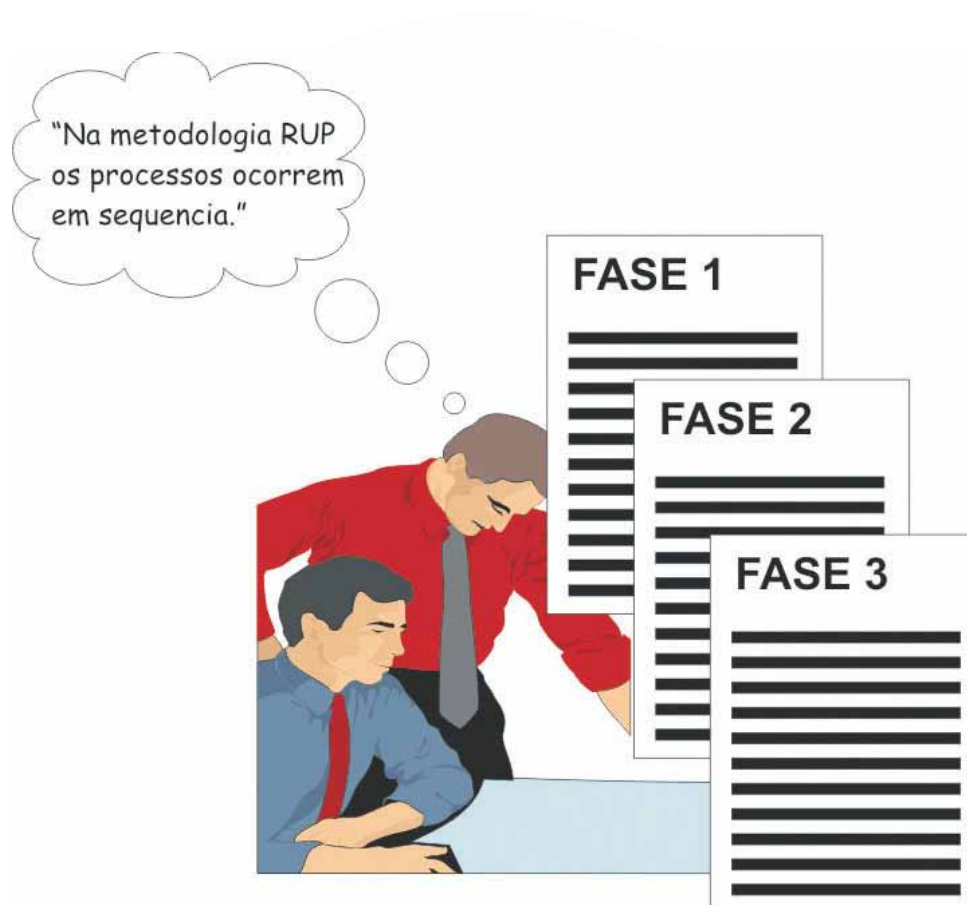


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

- **XP**

O Extreme Programming é uma metodologia geralmente usada por pequenas equipes de programadores. O ponto principal é a interação da equipe com o cliente e a entrega semanal de unidades de softwares prontas para uso. O XP incorpora nas suas premissas que o objetivo do software é mutável e o cliente pode mudar de ideia no meio do processo. As mudanças são bem-vindas e fazem parte do processo de criação.



ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

- **Scrum**

O método Scrum de desenvolvimento é utilizado em algumas empresas de construção de veículos no Japão e consiste na entrega de pequenos programas funcionais em intervalos regulares e reuniões diárias sobre o andamento.

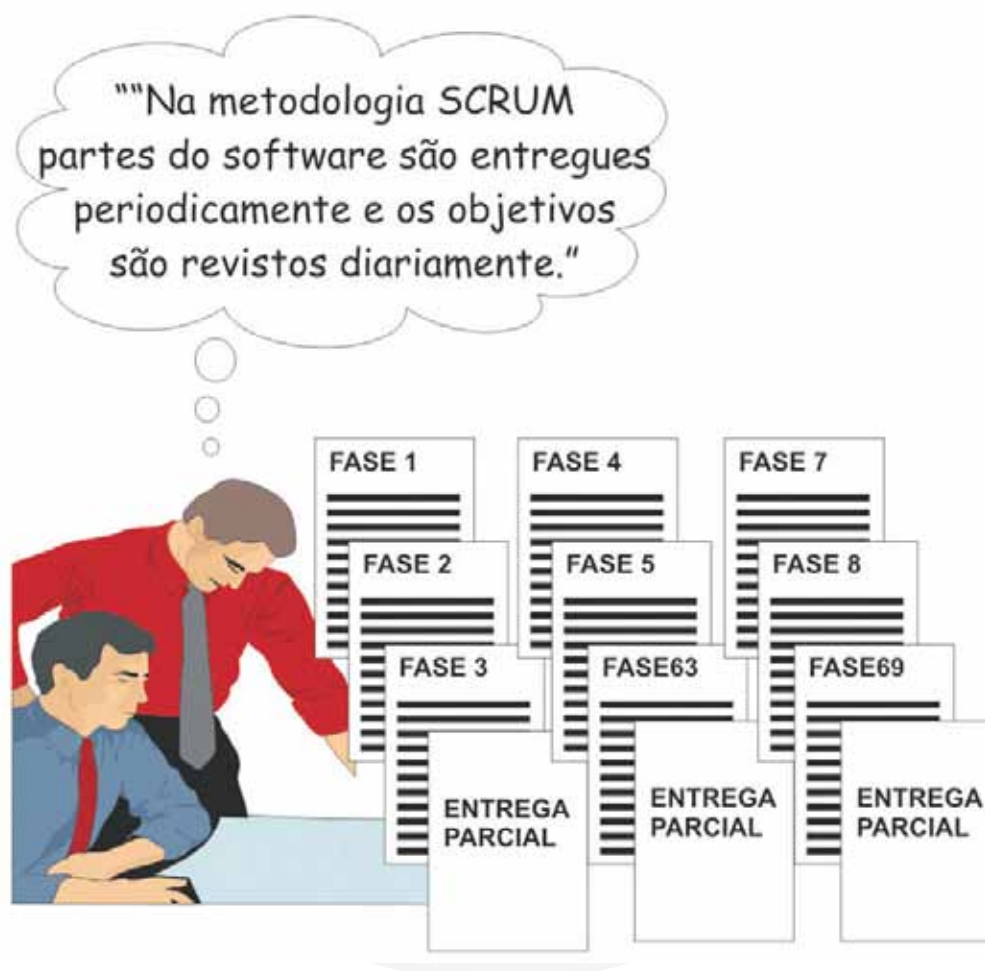


ILUSTRAÇÃO: EDUARDO JOSÉ DE SOUZA ENGELMANN

5.8. Resumo

Para desenvolver um software com sucesso, é necessário analisar vários aspectos que fazem parte desse ambiente. Escolher a linguagem, a tecnologia, a equipe e a metodologia corretas é um ponto crucial para o sucesso.

A seguir, temos um exemplo de um ambiente de desenvolvimento:

PROJETO LOJA ON LINE

1. Ambiente: ASP.NET
2. Linguagem: C#
3. Banco de dados: SQL Server
4. Ferramentas: Visual Studio, Photoshop e Flash
5. Framework: AJAX.NET
6. Profissionais: 1 analista, 2 programadores, 1 Web designer

PROJETO ORCAMENTO

1. Ambiente: Linux com ambiente gráfico
2. Linguagem: Java
3. Banco de dados: MySQL
4. Ferramentas: NetBeans
5. Profissionais: 1 programador



5

Teste seus conhecimentos

Ambientes de desenvolvimento de software

1. Qual o tipo de software que é chamado automaticamente pelo sistema operacional sem a intervenção do usuário?

- ☐ a) UI
- ☐ b) Componente
- ☐ c) Serviço
- ☐ d) Web Service
- ☐ e) Classe

2. Quando um componente está disponibilizado na Internet, como ele é chamado?

- ☐ a) Web Component
- ☐ b) Web Class
- ☐ c) Web Service
- ☐ d) WebMaster
- ☐ e) Web Object

3. Qual das alternativas a seguir não corresponde a uma linguagem de programação?

- ☐ a) VB
- ☐ b) Ruby
- ☐ c) Perl
- ☐ d) C#
- ☐ e) ASP.NET

4. Qual das alternativas a seguir não apresenta um banco de dados?

- ☐ a) SQL Server
- ☐ b) Oracle
- ☐ c) DB2
- ☐ d) T-SQL
- ☐ e) MySQL

5. Qual das alternativas a seguir apresenta um framework?

- ☐ a) Visual Studio
- ☐ b) SQL Server
- ☐ c) Struts
- ☐ d) Java
- ☐ e) VB Express

The background features a collection of 3D rectangular blocks in various colors (purple, blue, yellow, green, orange, pink, grey) arranged in a scattered, overlapping manner. Some blocks are solid, while others have a central square cutout. Thin, curved lines in blue, red, and yellow weave through the blocks, adding a sense of movement and connectivity. The overall aesthetic is clean, modern, and geometric.

Mãos à obra!

Ambientes de desenvolvimento de software

Laboratório 1

Para este exercício não existe uma única resposta certa e definitiva.

A – Treinando com ambientes de desenvolvimento de software

1. Complete o quadro a seguir:

PROGRAMA DE CONTROLE DE ESTOQUE

Linguagem de Programação: JAVA

Ferramenta para desenvolvimento: _____

Banco de Dados: SQL Server

CONTROLE ADMINISTRATIVO DE VENDAS

Linguagem de Programação: Visual Basic .NET

Ferramenta para desenvolvimento: _____

Banco de Dados: MySQL

PROGRAMA DE CONTROLE DE PROCESSOS PARA UMA FIRMA DE ADVOCACIA

As informações serão acessadas apenas dentro da empresa.

O Word é muito utilizado para criar cartas e processamentos.

Linguagem de Programação: _____

Ferramenta para desenvolvimento: Visual Studio

Banco de Dados: _____

Ambiente (Web ou Windows): _____