PROGRAMAÇÃO ORIENTADA A OBJETOS

Erick Guilhon (Org.)

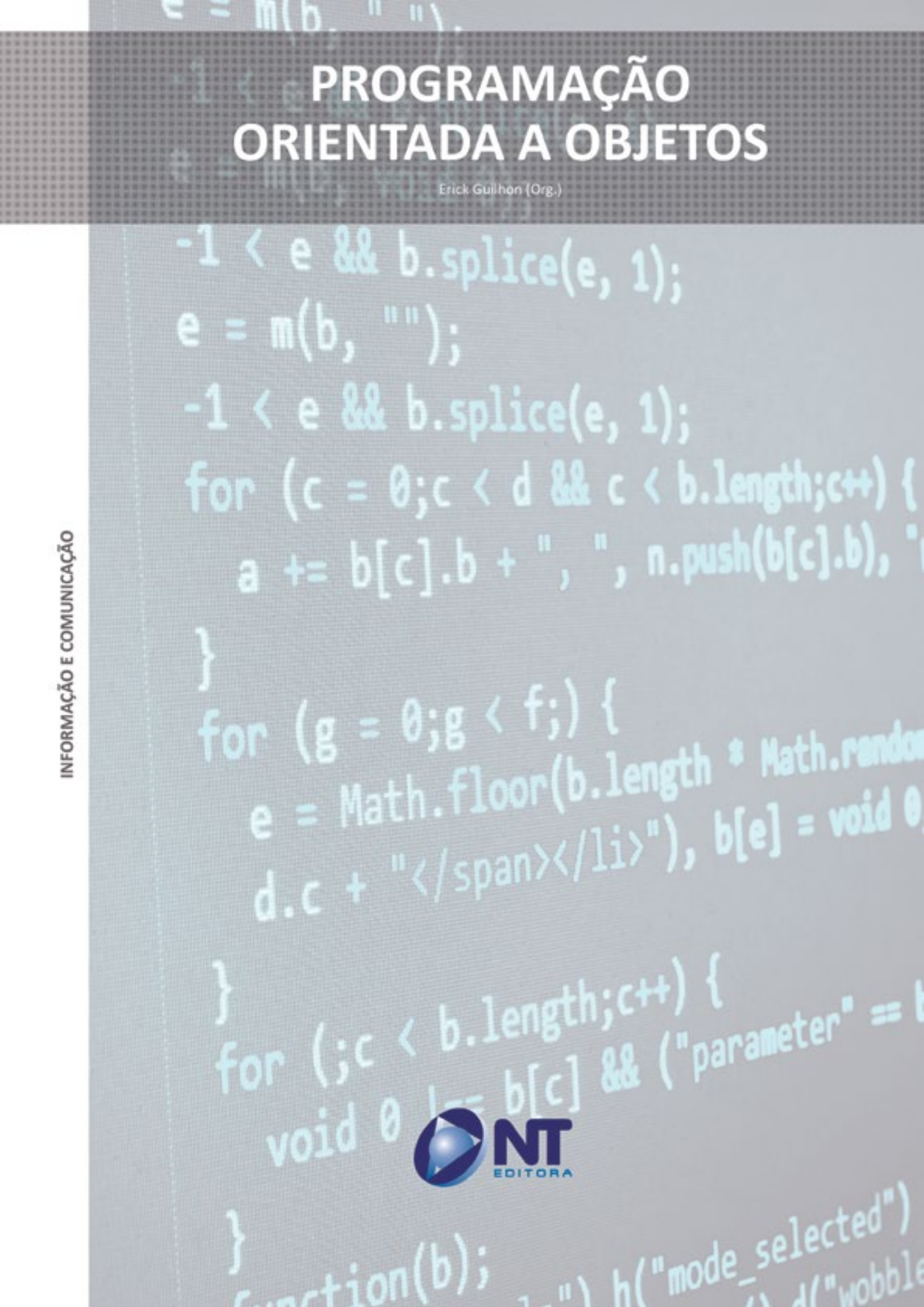
INFORMAÇÃO E COMUNICAÇÃO

NT EDITORA

PROGRAMAÇÃO ORIENTADA A OBJETOS

Erick Guilhon (Org.)

INFORMAÇÃO E COMUNICAÇÃO



NT EDITORA

Organizador
Erick Guilhon

Design Instrucional
NT Editora

Projeto Gráfico
NT Editora

Revisão
NT Editora

Capa
NT Editora

Editoração Eletrônica
NT Editora

Ilustração
NT Editora

NT Editora, uma empresa do Grupo NT
SCS Quadra 2 – Bl. C – 4° andar – Ed. Cedro II
CEP 70.302-914 – Brasília – DF
Fone: (61) 3421-9200
sac@grupont.com.br
www.nteditora.com.br e www.grupont.com.br

Guilhon, Erick (Org.).
Programação orientada a objetos / Erick Guilhon (Org.) – 1. ed. – Brasília: NT Editora, 2018.
160 p. il. ; 21,0 X 29,7 cm.

ISBN 978-85-8416-281-9

1. Programação. 2. Objetos. 3. Linguagem Java.
I. Título

ÍCONES

Prezado(a) aluno(a),
Ao longo dos seus estudos, você encontrará alguns ícones na coluna lateral do material didático. A presença desses ícones o(a) ajudará a compreender melhor o conteúdo abordado e a fazer os exercícios propostos. Conheça os ícones logo abaixo:

Saiba mais
Esse ícone apontará para informações complementares sobre o assunto que você está estudando. Serão curiosidades, temas afins ou exemplos do cotidiano que o ajudarão a fixar o conteúdo estudado.

Importante
O conteúdo indicado com esse ícone tem bastante importância para seus estudos. Leia com atenção e, tendo dúvida, pergunte ao seu tutor.

Dicas
Esse ícone apresenta dicas de estudo.

Exercícios
Toda vez que você vir o ícone de exercícios, responda às questões propostas.

Exercícios
Ao final das lições, você deverá responder aos exercícios no seu livro.

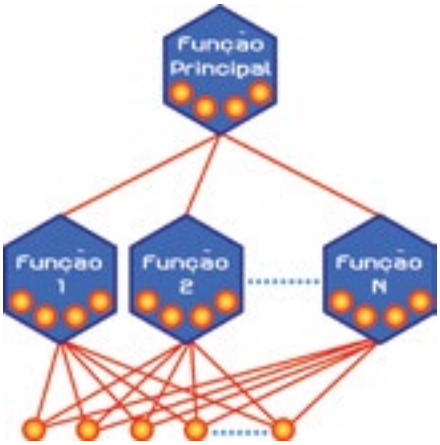
Bons estudos!

Sumário

1. INTRODUÇÃO

1.1 Decomposição funcional

Uma abordagem típica usada no desenvolvimento de programas complexos consiste em decompor os programas em diversos módulos e dividir cada módulo em diversas funções (Fig.1.01). Cada função é responsável por parte da solução do problema. Esta abordagem de desenvolvimento se baseia na decomposição funcional. Embora a decomposição funcional tenha sido amplamente utilizada nos últimos anos, apresenta algumas deficiências, tais como o fraco acoplamento entre dados e funções.



Fraco acoplamento: Cada função é responsável por parte da solução do problema. Esta abordagem de desenvolvimento se baseia na decomposição funcional, que, embora tenha sido amplamente utilizada nos últimos anos, apresenta algumas deficiências. Um tipo de deficiência facilmente identificável é o fraco acoplamento entre os dados e as funções.

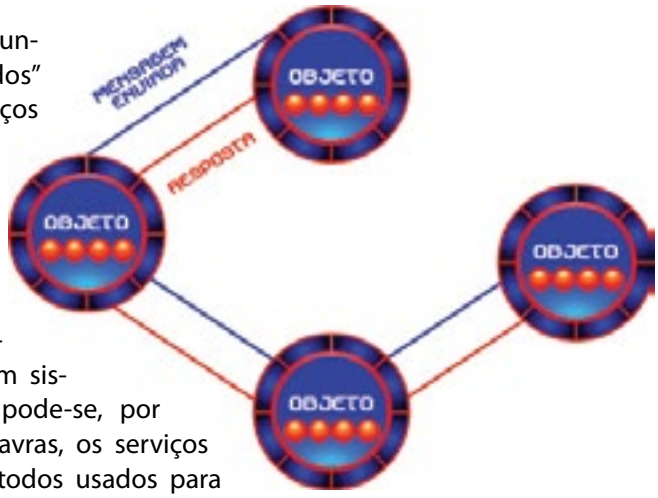
Fortemente acoplados: Em um programa desenvolvido usando-se orientação a objetos, existem facilidades para se definir quais as funções que podem acessar determinados dados. É fácil, portanto, limitar o acesso aos dados a apenas um conjunto determinado de funções. Por este motivo, diz-se que existe forte acoplamento entre dados e funções.

1.2 Orientação a objetos

A orientação a objetos é uma tecnologia de desenvolvimento composta por metodologias e linguagens usadas na análise, no projeto e na implementação de programas. Apesar de existir há décadas, só recentemente a orientação a objetos adquiriu popularidade. Em parte, esta popularidade decorre do uso crescente de interfaces gráficas e da arquitetura distribuída dos programas atuais, características que são adequadamente suportadas pela orientação a objetos.

Os programas desenvolvidos usando-se a orientação a objetos são compostos por módulos cujos dados e funções são fortemente acoplados (Fig.1.02). Estes módulos são denominados "obje-

tos" e, em cada um deles, temos dados e funções. As funções são denominadas "métodos" e contêm o código responsável pelos serviços providos pelos objetos. Os dados, por sua vez, só podem ser acessados por meio do código presente nos métodos do objeto (Fig.1.03). Por isso, costuma-se dizer que os dados são encapsulados.



Considere, por exemplo, um objeto que modela uma conta corrente em um sistema bancário. Por meio dos métodos, pode-se, por exemplo, depositar e sacar. Em outras palavras, os serviços **depositar** e **sacar** são providos pelos métodos usados para solicitar serviços aos objetos que modelam as contas bancárias. Além dos métodos responsáveis pelos serviços, cada objeto que modela uma conta bancária contém dados, como o nome do dono da conta e o valor do saldo. Como os dados normalmente não podem ser acessados diretamente, o acesso é feito por meio dos métodos.

O forte acoplamento entre dados e código facilita o desenvolvimento de programas modulares. A modularidade resultante facilita tanto a manutenção e a evolução dos programas quanto a reutilização de código entre programas destinados a uma mesma área de aplicação. A possibilidade de se desenvolver programas a partir de módulos existentes reduz o custo de desenvolvimento, sendo uma das vantagens da orientação a objetos.

Fig.1.03 – Encapsulamento dos dados

Exercitando o conhecimento...

Leia com atenção cada uma das afirmações abaixo e identifique-as como **Verdadeira** ou **Falsa**.

1 – Nos módulos, estão contidos os códigos responsáveis pelos serviços providos pelos objetos.

( ) Verdadeira

( ) Falsa

2 – Os dados só podem ser acessados através do código presente nos métodos do objeto. Por isso, costuma-se dizer que os dados estão encapsulados.

( ) Verdadeira

( ) Falsa

1.3 Fases de desenvolvimento

O desenvolvimento de um programa orientado a objetos é realizado em fases: análise, projeto e implementação (Fig.1.04). A primeira fase é a análise, na qual é feita a especificação dos requisitos do programa que é descrito com base nas expectativas dos usuários. Ao final, devem ser removidas as omissões da especificação inicial e os usuários devem estar convencidos de que a especificação descreve o programa desejado. As principais atividades da análise são:

- levantar a especificação junto aos usuários;
- descrever o programa por meio de casos de uso;
- identificar objetos no domínio do problema;
- identificar, para cada objeto, suas responsabilidades;
- identificar os relacionamentos entre os objetos.

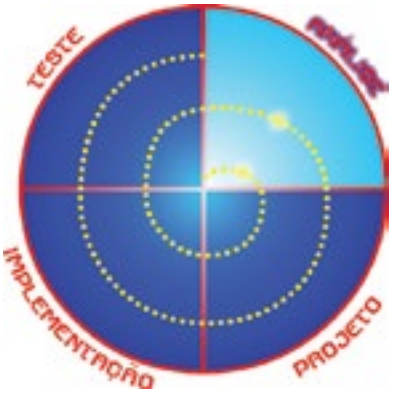


Fig.1.04 – Fases do desenvolvimento

A partir do momento em que os aspectos de implementação começam a influenciar os modelos, tem início a fase de projeto. Nessa fase, os modelos definidos na análise são adaptados ao ambiente no qual o programa será implantado. Os modelos resultantes da fase de projeto descrevem o programa da forma como o mesmo será realmente implementado. As principais atividades do projeto são:

- identificar aspectos do ambiente de implementação e investigar as consequências de tais aspectos sobre o programa a ser implementado;
- traduzir os modelos gerados na fase de análise de forma a escreverem um programa possível de ser implantado, levando em consideração as restrições impostas pelo ambiente de implementação;
- identificar os objetos e como estes interagem.

A fase seguinte no desenvolvimento de um programa é a implementação. No caso de um programa orientado a objetos, a implementação torna-se mais simples se for usada uma linguagem de programação que suporte os conceitos presentes na orientação a objetos. Com tais linguagens, é possível mapear os conceitos usados na análise e no projeto para os códigos dos programas. Os principais conceitos que devem ser suportados são: classes (tipos de objetos), objetos, polimorfismo e herança. Vale salientar que a orientação a objetos é uma tecnologia. O simples uso de uma linguagem de programação adequada não é suficiente para garantir que os benefícios desta tecnologia sejam alcançados.

1.4 Linguagens de programação

No que diz respeito ao suporte à orientação a objetos, as linguagens de programação são classificadas como convencionais, baseadas em objetos ou orientadas a objetos. As linguagens convencionais e as baseadas em objetos não suportam integralmente os conceitos presentes na orientação a objetos. São linguagens convencionais: C, Pascal, Cobol, Fortran e Basic. São linguagens baseadas em objetos: ADA e Modula-2.

As linguagens orientadas a objetos suportam integralmente os principais conceitos presentes na orientação a objetos. Os principais conceitos suportados por tais linguagens são: classes (tipos de objetos), objetos, polimorfismo e herança. São linguagens orientadas a objetos: C++, Object Pascal, Smalltalk e Java.

1.5 Testes

Ao longo do desenvolvimento de um programa orientado a objetos, é fundamental que seja dada atenção ao planejamento e à execução dos testes a fim de se garantir a qualidade do programa. Os testes devem ser parte integrante do desenvolvimento, não devendo ser realizados apenas ao final da implementação. Os testes são geralmente classificados como: testes das unidades, testes de integração e testes do sistema.

Os testes das unidades têm por objetivo testar o funcionamento individual de cada objeto e os serviços por eles providos. Os testes de integração verificam agrupamentos de objetos, denominados subsistemas e os testes do sistema, por fim, testam o sistema como um todo. Os testes de integração e do sistema verificam se o programa atende aos casos de uso descritos na especificação, submetendo-o a diferentes cargas de uso e analisando aspectos relativos à performance.

1.6 Metodologias

O desenvolvimento de um programa orientado a objetos pode se tornar uma tarefa mais simples se for adotada uma metodologia de desenvolvimento. As metodologias descrevem técnicas para facilitar e organizar o desenvolvimento de programas e geram, como resultado, diversos modelos. Cada modelo apresenta uma visão do sistema: visão de dados, de comportamento e da arquitetura. Os modelos apresentam um nível crescente de detalhamento dependendo da fase em que se encontra o desenvolvimento.

A visão de dados, representada por meio de diagramas de classes e diagramas de objetos (Fig.1.05), descreve os tipos de objetos e como estes se relacionam. A visão de comportamento, representada, por exemplo, por meio de diagramas de interação (Fig.1.06), descreve o comportamento dinâmico do sistema. A visão de arquitetura descreve, por exemplo, a divisão do sistema em subsistemas e pode assumir diferentes formas, dependendo da metodologia utilizada (Fig.1.07).

Fig.1.05 – Diagrama de classe e de objetos

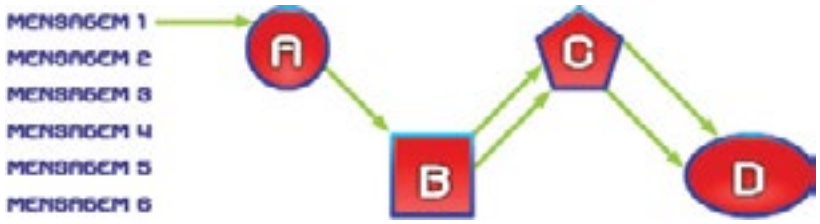


Fig.1.06 – Diagrama de interação

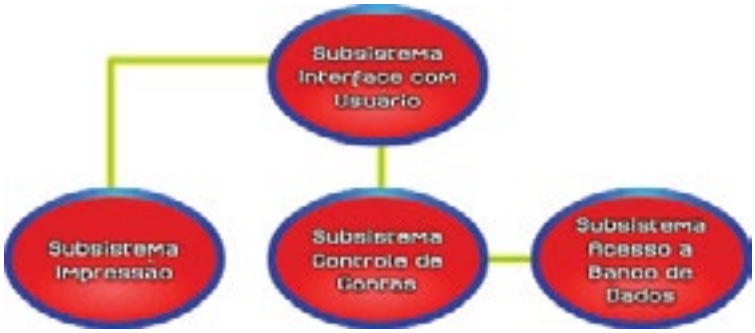


Fig.1.07 – Diagrama da arquitetura

As metodologias de desenvolvimento são geralmente compostas por um processo e por uma linguagem de modelagem (Fig.1.08). O processo descreve aspectos como: fases de desenvolvimento, documentos resultantes de cada fase, conselhos para a identificação dos objetos, etc. A linguagem representação dos modelos que descrevem o programa em desenvolvimento e merece destaque é a Unified Modeling Language (UML). Dentre as diversas metodologias para análise e projeto de programas orientados a objetos podemos destacar:

- Object Oriented Design;
- Object Oriented Systems Analysis;
- Object Modeling Technique;
- Object Oriented Analysis;
- Hierarchical Object Oriented Design;
- Object Oriented Structured Design;
- Responsability Driven Design;
- Object Oriented Role Analysis; e
- Object Oriented Software Engineering.

As metodologias para o desenvolvimento de programas orientados a objetos apresentam diversas similaridades, por exemplo:

- ao longo das fases de análise e projeto são mantidos integrados os dados e os comportamentos dos objetos;
- suportam classes, objetos e herança;
- sugerem como identificar classes e suas responsabilidades;
- sugerem como identificar características semelhantes entre classes e como agrupar classes em hierarquias;
- sugerem como identificar as interações entre objetos.

Parabéns, você finalizou esta lição!

Agora responda às questões ao lado.

Exercícios

Questão 01 – Com relação à Orientação de objetos, marque a afirmativa falsa:

a) possibilita o forte acoplamento entre dados e funções;

b) é uma tecnologia que se popularizou recentemente;

c) possibilita o desenvolvimento de programas modulares;

d) não facilita a reutilização do código.

Questão 02 – A respeito dos objetos, marque a afirmativa incorreta:

a) são compostos por dados e funções;

b) funções são chamadas métodos;

c) dados podem ser acessados a partir de qualquer função;

d) métodos definem uma interface para o objeto.

Questão 03 – Dentre as alternativas abaixo, marque aquela que não é responsabilidade da fase de análise:

a) levantar a especificação junto aos usuários;

b) especificar aspectos da plataforma de implementação;

c) identificar objetos no domínio do problema;

d) identificar omissões na especificação inicial.

Questão 04 – Dentre as alternativas abaixo, marque aquela que não é responsabilidade da fase de projeto:

a) elaborar especificação inicial;

b) levar em consideração aspectos da plataforma de implementação;

c) identificar objetos que serão codificados na implementação;

d) usar os modelos definidos na fase de análise.

Questão 05 – marque a afirmativa que não indica uma característica de uma linguagem orientada a objetos:

a) suporte e herança;

b) suporte a classes e objetos;

c) suporte a polimorfismo;

d) suporte a variáveis globais.

Questão 06 – marque a opção que não indica uma linguagem orientada a objetos:

a) Object Pascal;

b) Java;

c) C++;

d) C.

Questão 07 – A respeito de metodologias, marque a afirmativa falsa:

a) compostas por processos e linguagens de modelagem;

b) UML é um processo de modelagem;

c) Booch, Rumbaugh e Jacobson são famosos metodologistas;

d) existem diversas formas de metodologias de modelagem orientadas a objetos.

Questão 08 – Marque a afirmativa falsa a respeito de metodologias orientadas a objetos:

a) mantêm integrados os dados e o comportamento;

b) suportam como identificar a integração entre objetos;

c) sugere como identificar classes e agrupá-las em hierarquias;

d) usadas apenas na fase de projeto.

Questão 09 – Marque a afirmativa falsa a respeito de modularidade:

a) facilitada na orientação a objetos pelo encapsulamento;

b) reduz o tempo de desenvolvimento e facilita a manutenção;

c) nos módulos que compõem um programa orientado a objetos, dados e códigos são fracamente acoplados;

d) é possível o desenvolvimento de programas não orientados a objetos que sejam modulares.

Questão 10 – Marque a afirmativa falsa sobre testes:

a) podem ser de unidade, de integração ou de sistema;

b) testes de unidades testam o funcionamento individual de cada objeto;

c) testes de integração testam o sistema como um todo;

d) testes de sistema verificam se o programa atende aos casos de uso especificados.