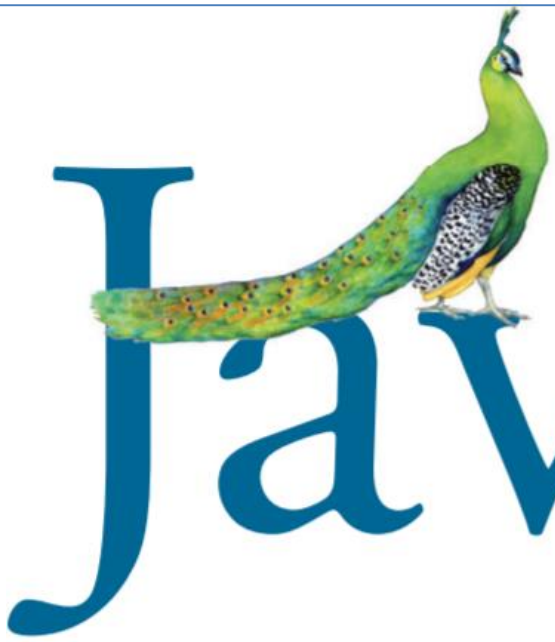
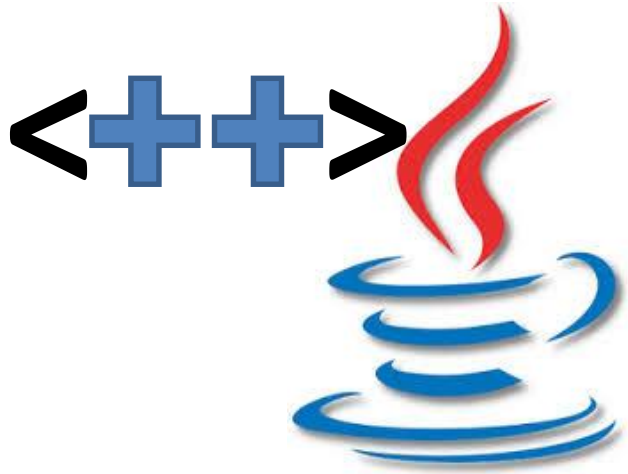
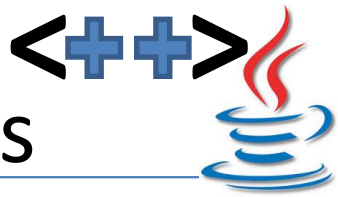


Baixe este projeto em Java no Github.

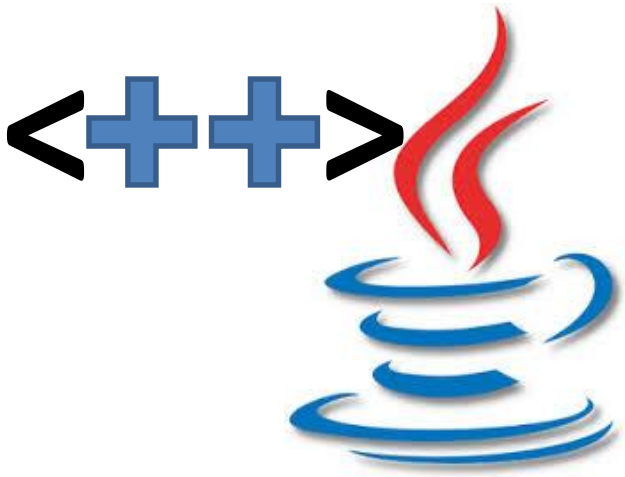
Programação Orientada a Objetos



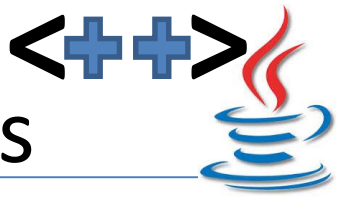
Programação Orientada a Objetos



1 - Introdução à orientação a objetos.
2 - Linguagens típicas orientadas a objetos.
3 - Programação orientada a objeto em Java.



Programação Orientada a Objetos



Autor

Cleilson Henrique de Araujo

Prefácio

A informação e a sobrevivência

O aprendizado atual está como sempre foi: manipulado, desde sua criação.
Desde o tempo dos homens e mulheres das cavernas a informação é sobrevivência. Quando o ser humano passou da fase de utilizar a informação para apenas sobreviver, ele começou a utilizá-la como meio de controle e repressão a sua própria tribo/sociedade/meio em que vive/natureza.

Desde então temos tribos contra tribos, civilizações contra civilizações, impérios contra impérios...
Sendo que o que determina o vencedor desde o tempo das cavernas até hoje é a informação.

Quando o ser humano começou a utilizar a primeira grande informação que foi a utilização do fogo, **a natureza** e as outras tribos atacadas não tinham chance alguma e foram derrotadas.

Quando o ser humano começou a utilizar armas primitivas do tempo da pedra lascada de fabricação humana, as outras tribos atacadas não tinham chance alguma e foram derrotadas.

Quando o ser humano começou a utilizar armas, utilizando o minério de ferro entre outros, posteriormente manipulando-os até se chegar as espadas lanças..., as outras tribos atacadas não tinham chance alguma e foram derrotadas.

Quando o império chinês criou e utilizou a pólvora e o seu uso em combates/guerras se espalhou as sociedades que utilizavam apenas espadas lanças... foram derrotadas. E chegamos a criação da bomba atômica e nuclear, como saber se o aprendizado é manipulado?
Tente você(uma sociedade/grupo/indivíduo) criar uma bomba atômica ou nuclear.

Nas nossas aulas, não vou lhe ensinar a criar uma bomba atômica ou nuclear. Irei repassar as informações acumuladas das gerações anteriores para a sobrevivência das futuras gerações.

Conforme o nosso histórico, a geração ou gerações que irão sobreviver serão as que mais bem utilizarem a informação.

Livro/Vídeo

Informações complementares

Links das vídeo aulas online:

www.youtube.com/c/JMarySystems/playlists?view_as=subscriber

Dos conteúdos deste livro na web:

http://www.jmarysystems.com.br/Leitura_e_Aprendizagem/Leitura_e_Aprendizagem.html

Das perguntas e respostas deste livro na web:

http://www.jmarysystems.com.br/Perguntas_e_Respostas/Perguntas_e_Respostas.html

Do programa em Java e outros projetos que além do conteúdo do livro, tem uma bateria de exercícios que complementam a totalidade da aprendizagem:

https://github.com/jmarysystems?tab=repositories

Visite-nos na web:

http://www.jmarysystems.com.br

jmarysystems@mail.com

85.98193.1133

Índice

Unidade 1

Programação Orientada a Objetos

Símbolos utilizados


- Neste conteúdo


Para início de conversa


Orientação da disciplina


Fique atento


Indicação de livro


Palavras do professor

Veja o vídeo!


Subcategoria


Assunto


Exemplo(s)


Fica a dica


Categoria

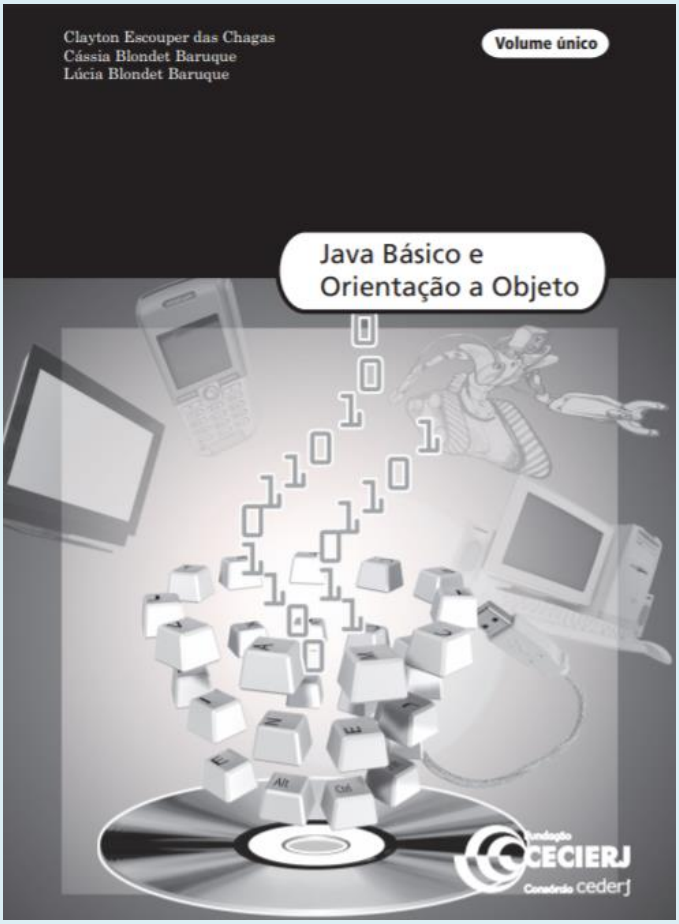

Subcategoria


Exercício(s)

Referências Bibliográficas

Indicação de livro

Título: Java Básico e Orientação a Objeto



Autor(es): Clayton Escouper das Chagas; Cássia Blondet Baruque; Lúcia Blondet Baruque

Data: 17/03/2011

Tipo: Documento

Baixar PDF: https://canal.cecierj.edu.br/recurso/7007

Programação Orientada a Objetos

Construção do conhecimento

- Introdução

Para início de conversa

Vamos começar a construção do conhecimento neste oceano de letras, sílabas, palavras, frases e versos que bem estruturados produzem e transmitem conhecimento.

Para aprender, uma boa leitura ou várias até fixar o que foi lido é o primeiro passo no mundo do conhecimento. O segundo passo é a resolução dos exercícios escritos à mão e no mesmo dia estudado.

Orientação da disciplina

Nesta disciplina ou qualquer outra, você precisa absorver o que estuda.

Por isso após você estudar todo ou parte do conteúdo desta disciplina, resolva os exercícios ou crie alguns você mesmo com suas anotações.

Lembrando que o estudo é feito após assistir a aula, assistir a aula não é estudar.

Todo conteúdo após estudado é necessário ser feito a resolução dos exercícios para poder fixar o conhecimento.

No entanto os exercícios tem de serem escritos e resolvidos no caderno à mão, após o estudo do conteúdo e no mesmo dia.

Caso se interesse mais sobre estas táticas condicionadas a nós seres humanos criadas, desenvolvidas e aperfeiçoadas durante décadas, assista a palestra do gênio e super.:

Prof Pierluigi Piazzi

Link: https://www.youtube.com/watch?v=BoMmj_Xt-pk

Construção do conhecimento

- Introdução

Fique atento

Não se esqueça que em cada unidade você encontrará um vídeo que deverá ser assistido no momento indicado.

Este vídeo irá complementar este conteúdo a fim de ampliar, ainda mais, o seu conhecimento acerca do tema tratado em cada unidade.

Ao final de cada unidade você deverá realizar as atividades.

Fica a dica

Uma pequena observação que quero deixar para você é:

caso não tenha visto por completo a Unidade anterior, é importante que você retorne e estude totalmente essa unidade.

A nossas disciplinas são construídas em cima de uma sequência lógica de assuntos.

Todos eles têm como objetivo desenvolver o seu raciocínio lógico.

Isto significa que se você avançar de unidade sem concluí-la, pode ter seu desenvolvimento comprometido.

Orientação a objetos

Para mostrarmos a importância do aprendizado da linguagem de programação Java, precisamos antes fazer uma retrospectiva sobre programação e suas linguagens através dos tempos, ate chegarmos em Java.

O nascimento da computação, na década de 40 e depois da 2º guerra, o computador era apenas uma máquina pouco amigável.

A única forma de se comunicar com ele era através de linguagem de máquina – um tipo de programação de baixo nível – exclusiva para a máquina que estava sendo programada.

Essa linguagem consistia de comandos binários ou hexadecimais, e isso era extremamente trabalhoso e improdutivo do ponto de vista da programação, era uma verdadeira “tripa” de comandos em linguagem de máquina.

Estes comandos eram feitos por chaveamentos em instruções binárias de máquina (código de máquina ou linguagem de máquina), usando-se o painel frontal com várias chaves que eram usadas para dar comandos ao computador, entrado esses códigos binários. Enquanto os programas continham somente algumas centenas de instruções, esse método funcionava.

A implementação de qualquer tarefa ou suas modificações em linguagem de máquina era extremamente penosa. Com o tempo, percebia-se que o programador gastava muito do seu trabalho se digladiando com comandos e instruções de máquina e pouco com a lógica do seu sistema.

Um aluno de hoje pode achar uma loucura o que vou falar, mas, na evolução seguinte, para os Assemblers, a vida do programador melhorou muito.

Foi uma mudança do purgatório para o paraíso, eram as chamadas Linguagens de Segunda Geração onde o programador podia usar nomes simbólicos para seus comandos e dados.

Orientação a objetos

Na década de 1960 e início da de 1970 Os computadores ainda tinham pouca memória e poucos recursos, mas as linguagens de “alto nível” já eram mais acessíveis e a programação em linguagem de máquina ou Assembler eram somente complementares.

No IBM 1130 a linguagem principal era o Fortran. As linguagens de “alto nível” como FORTRAN e COBOL, eram chamadas de Linguagens de Terceira Geração, pois traziam um novo paradigma ao se libertarem da dependência dos códigos de máquina e favoreciam a descrição lógica dos procedimentos da programação.

Durante esse tempo, na academia, as linguagens de programação estavam sendo formalizadas, em termos de suas gramáticas, sintaxes etc.

Uma primeira dessas linguagens mais bem-nascidas foi a Algol 60 (Algorithmic Language). Outras linguagens dessa mesma época tiveram algum destaque em nichos específicos como, por exemplo, Simula, Lisp, Pascal, C, entre outras.

Também na década de 1970 veio a onda da programação estruturada, dando uma nova perspectiva para a forma de programar com mais qualidade.

Entretanto, na prática, nesse tempo se programava muito em Cobol, Fortran e depois em PL/I (uma linguagem que era um “mix” de Fortran, Cobol e Algol).

Mais tarde apareceu a linguagem BASIC que ficou famosa nos primeiros microcomputadores. BASIC é o acrônimo de Beginners Allpurpose Symbolic Instruction Code, e nasceu de Fortran (acrônimo de Formula Translation).

Orientação a objetos

BASIC deveria ser a linguagem didática para iniciantes, mas não trazia nada de bom para programação organizada, ou seja, estruturada.

Costumo dizer que quem aprendeu BASIC como sua primeira linguagem tem “problema de infância”.

Contudo, seu sucesso foi muito grande. Todos programavam em BASIC, tanto que até hoje temos linguagens originadas a partir dela, mesmo sendo as versões atuais muito diferentes de sua antepassada, como é o caso de Visual BASIC da Microsoft.

O importante nicho da área comercial, ou seja, as áreas bancária, atuária, seguros, comércio, entre outras que tinham a necessidade de utilizar recursos computacionais face à grande quantidade de informações e manipulação de dados que concentravam, ficou com COBOL.

Hoje, muitos dos sistemas legados dessas áreas estão escritos nessa linguagem.

Com o passar do tempo, as Linguagens de Terceira Geração foram incluindo mais facilidades para a construção de programas bem modularizados e estruturados.

A programação com essas linguagens foi difundida nos diversos computadores, que deixaram de ser luxo das grandes indústrias, universidades e órgãos militares e se tornaram muito mais capilares, tendo seu ápice com a criação e disseminação dos computadores pessoais (PCs).

Com o avanço tecnológico das linguagens, foi necessário facilitar o desenvolvimento de sistemas, já que a demanda crescera desesperadamente.

Dessa forma, os paradigmas evoluíram e as técnicas de análise e projeto de sistemas se tornaram cada vez mais refinadas.

Orientação a objetos

Portanto, as linguagens que começaram com instruções de máquina e Assemblers deram lugar às estruturadas.

As linguagens estruturadas evoluíram sempre focadas na legibilidade e elegância de programação.

Também trouxeram as facilidades de funções e sub-rotinas ou procedimentos e de utilização de bibliotecas para a modularização de sistemas.

Em paralelo, a área de programação evoluía com novas técnicas e métodos para melhorar a qualidade dos sistemas produzidos. Isso tudo resultou em uma nova engenharia (Engenharia de Software).

Esta necessidade de melhoria trouxe mais um conceito, que apesar de ter nascido na década de 1970, só tomou forma e cresceu na década de 1980, e até hoje domina o projeto de qualquer linguagem que seja criada com o intuito de concorrer com as outras: a Orientação a Objetos.

Costumo dizer que, com esse paradigma de programação, estamos chamando a atenção da importância da modelagem dos dados junto com seu comportamento.

Segundo esse paradigma, o programa implementa as classes de objetos que melhor representam o enunciado do problema.

Assim, os sistemas seriam projetados de forma mais intuitiva, retratando o mundo tal qual ele é.

Duas das primeiras linguagens desse novo paradigma, que se popularizaram foram C++ e Smalltalk. Essas linguagens introduziram o conceito de classes de objetos em hierarquias. A programação ficou mais hierarquizada e com isso surgiram novos conceitos e características que ajudaram muito na programação, garantindo-lhe mais elegância e simplicidade.

Orientação a objetos

A partir deste momento, o programador poderia se dedicar realmente a expressar sua lógica de negócios.

Conceitos como herança, polimorfismo, encapsulamento e outros são até hoje parte principal da teoria da Orientação a Objetos em qualquer linguagem de programação desse gênero.

No entanto, cabe ressaltar que o conceito de orientação objeto depende mais da mentalidade do programador do que da linguagem de programação que está sendo utilizada.

Pode-se conseguir um programa razoavelmente orientado a objeto em linguagens tipicamente estruturadas, assim como se pode conseguir um programa estruturado em linguagens com sintaxe e recursos de orientação a objetos.

Paralelamente, um outro movimento tecnológico ganhou muita força na mesma época da consolidação da Orientação a Objetos como paradigma de programação: o mundo da computação nunca mais foi o mesmo depois do nascimento da web.

Mas com ela também surgiram outros problemas, como a interoperabilidade das plataformas, já que os sistemas numa rede mundial são os mais heterogêneos possíveis.

Para resolver isto, emergiram os estudos sobre máquinas virtuais. A Sun Microsystems tinha um projeto inovador em uma linguagem denominada Oak, cujo slogan era: write once, run anywhere, (em português: escreva uma vez, rode em qualquer lugar).

O novo contexto justificava o retorno desse projeto com força, e essa linguagem que trazia uma plataforma associada a ela mudou seu nome, devido a questões de direitos autorais, para o que seria a plataforma mais promissora do fim do século XX: **Java**.

Orientação a objetos

Java podia ser executada em qualquer plataforma, era compilada uma única vez, numa única plataforma, gerando um código intermediário chamado bytecode, e este poderia ser interpretado por uma Máquina Virtual Java (JVM) em qualquer tipo de dispositivo, seja um computador, um celular, uma televisão ou até uma geladeira.

Depois desse fenômeno os conceitos de programação nunca mais foram os mesmos, desde a década de 90, quando surgiu o Java, uma nova linguagem simples, elegante, orientada a objeto e portátil.

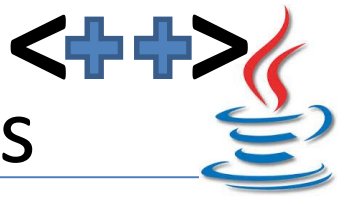
Orquestrada por uma empresa poderosa chamada Sun Microsystems tem toda uma comunidade participativa e comprometida com os mais modernos conceitos de uma linguagem de programação associada à engenharia de software.

Essa força do Java se mostrava mais poderosa ainda quando íamos para o mundo "www", onde as limitadas e sem graça páginas html ganharam efeitos muito mais interessantes através de objetos dinâmicos Java chamados Applets.

E com o tempo, a própria Sun viu a necessidade de abrir o Java, delegando-o completamente à sua comunidade, sempre organizada e rigorosa nos seus processos evolutivos.

Através de seus vários comitês, a padronização virou um conceito forte dentro do Java, o que permitiu que este sobrevivesse e continuasse crescendo até hoje, mesmo sem ter um dono.

Dessa forma, evoluímos das sub-rotinas e simples organização de programas para bibliotecas de classes, onde enfatizamos o reuso através de organizações mais elaboradas dessas bibliotecas nas APIs e Frameworks.



Orientação a objetos

Não se escreve mais código a partir do zero, mas se reusam as classes especializadas em seus contextos.

O desenvolvedor deixou de ser um programador braçal para ser um compositor, um projetista.

E com isso ganhou-se muito mais velocidade e consistência de desenvolvimento, o que é necessário, pois a demanda é grande, e continua crescendo.

E assim continua a saga do Java, elegante, simples, modular, mas ao mesmo tempo abrangente e completo.

Java é uma das linguagens mais importantes da atualidade, apesar de sua pouca idade (seu primeiro release 1.0 é de 1995) e não é mais uma promessa, mas uma realidade e com um futuro ainda muito longo e brilhante.

Nesse contexto, com uma linguagem que converge tantos aspectos fundamentais da computação, desde os conceitos de linguagem de programação e orientação a objetos, passando por engenharia de software e redes, vale ressaltar a importância do ensino de Java, onde este livro passa a ter papel importante pela forma didática que aborda o assunto, buscando o equilíbrio através de um conteúdo abrangente e completo, mas sem tirar a importância do cunho prático que deve nortear qualquer livro de programação.

Enfim, tenho certeza que qualquer aluno que queira aprender a linguagem de programação Java, qualquer que seja o seu nível, vai agregar conhecimentos sólidos ao ler esse livro sobre esta tecnologia que está dominando os mercados de programação em várias áreas do desenvolvimento: web, celulares, TV digital, robótica, entre outras.

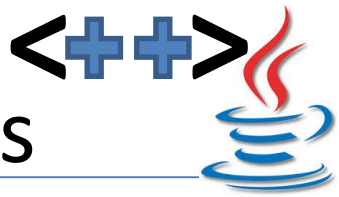
Prof. Dr. Rubens Nascimento Melo PUC-Rio

Orientação a objetos

Fica a dica

Programação orientada a objetos é a programação implementada pelo **envio de mensagens a objetos**.

Cada objeto irá responder às mensagens conhecidas por este, e cada objeto poderá enviar mensagens a outros, para que sejam atendidas, de maneira que ao final do programa, todas as mensagens enviadas foram respondidas, atingindo-se o objetivo do programa.



Exemplo(s)

Para tentar exemplificar as duas abordagens, tomemos como exemplo a frase:

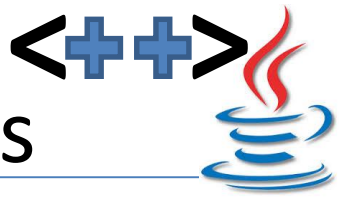
"O caminhão estaciona na fábrica e carrega sua mercadoria."

Vejamos, se analisássemos esta frase usando uma abordagem estruturada, pensaríamos logo em como o caminhão estacionaria na fábrica e como ele faz para carregar sua mercadoria, ou seja, pensaríamos na ação que está sendo feita (que na frase é representada pelos verbos) para transformá-la em procedimento ou funções.

Em orientação objeto, o **enfoque** com que se encara a frase é diferente: primeiro pensaríamos no objeto caminhão, no objeto fábrica e no objeto mercadoria, pensando como eles seriam e procurando definir seu comportamento, suas ações.

Após isto, é que pensaremos em como o caminhão se relaciona com a fábrica e com a mercadoria, e como a fábrica se relaciona com a mercadoria.

De modo simplificado, podemos dizer que ao analisarmos uma frase pensando de forma estruturada, damos ênfase aos verbos, e pensando orientado a objetos, damos ênfase aos substantivos.



Palavras do professor

Veremos a seguir algumas das vantagens que motivam veteranos programadores a readaptar-se para o paradigma de orientação a objeto:

• Sensível redução no custo de manutenção do software.
• Aumento na reutilização de código. Isso para não falar de outros. De fato, podemos ver que os princípios da engenharia de software podem ser aplicados de forma mais adequada quando usamos uma abordagem orientada a objetos.
• Redução no custo de manutenção: Na programação orientada a objetos, existem certas características, como herança e encapsulamento, que permitem que, quando for necessária alguma alteração, modifique-se apenas o objeto que necessita desta alteração, e ela propagar-se-á automaticamente as demais partes do software que utilizam este objeto.

O detalhadamente do motivo desta propagação, assim como os conceitos de herança e encapsulamento serão vistos mais adiante.

• Aumento na reutilização de código: Podemos dizer de modo simplificado, que o conceito de orientação objeto fornece a possibilidade de um objeto acessar e usar como se fossem seus os métodos e a estrutura de outro objeto.

Desta forma, quando, por exemplo, existirem **dois objetos bastante semelhantes**, com mínimas diferenças, **pode-se escrever os métodos apenas uma vez e usá-los para os dois objetos**.

Apenas os métodos que realmente forem diferentes para os dois objetos é que precisam ser escritos individualmente.

Isso é possível através do mecanismo de herança que, como eu já disse, será detalhado mais tarde.

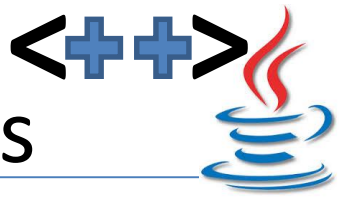
Orientação a objetos

Indicação de livro

Gostaria de recomendar a leitura dos capítulos que falam sobre os conceitos e princípios de orientação a objetos que se encontram nos livros de Engenharia de Software e de Análise e Projeto de Sistemas.

- SOMMERVILLE, I. Engenharia de Software.
- PFLEEGER, S. L. Engenharia de Software: Teoria e Prática.

Para facilitar a compreensão desses conceitos e a diferença nas duas abordagens mencionadas (estruturada e orientada a objetos), colocarei aqui um problema para ser resolvido através de uma implementação estruturada e também orientada a objetos.



Exemplo(s)

Exemplo de problema:
A imobiliária “MEU TERRENO”, é uma empresa que se especializou em vendas de terrenos em loteamentos onde os lotes são demarcados na forma retangular.

Essa imobiliária pretende desenvolver um programa que faça o cálculo dos valores dos terrenos comercializados.

Para isso é necessário fornecer alguns dados para o cálculo.

O objetivo desse programa é:
com base na largura e profundidade de um terreno retangular, assim como o valor do metro quadrado desse loteamento, calcular e exibir a área e o valor desse terreno.

Solução:
Basicamente, precisamos ler do teclado os dados para efetuar os cálculos.

O usuário irá fornecer a largura, profundidade e valor do metro quadrado da região.

Com base nesses dados o programa calcula e exibe a área do terreno, assim como o valor do mesmo. Veremos os códigos de duas abordagens para você ter uma ideia de um programa orientado a objetos. Será visto logo em seguida a versão do programa estruturado e depois a versão do mesmo programa orientado a objetos.

Para o usuário, quando ele for executar o programa, não terá nenhuma diferença na interface. Ele não teria como saber de que forma o programa foi codificado.

Orientação a objetos

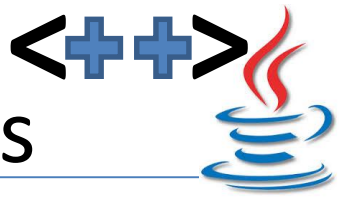
Exemplo(s)

Versão da solução implementada de forma estruturada:

Programa_Estruturado.java

```
1  import java.util.Scanner;
2
3  public class Programa_Estruturado {
4
5
6          public static void main ( String [ ]  args ) {
7                  // Programa usando programação Estruturada
8                  double largura , profundidade , valor_m2 , valor ;
9                  Scanner leia = new Scanner ( System.in ) ;
10                 System.out.print ( " Digite o valor da largura do terreno: " );
11                 largura = leia.nextDouble ( ) ;
12                 System.out.print ( " Digite o valor da profundidade do terreno: " );
13                 profundidade = leia.nextDouble ( ) ;
14                 System.out.print ( " Digite o valor do metro quadrado do terreno: " );
15                 valor_m2 = leia.nextDouble ( ) ;
16                 valor = largura * profundidade * valor_m2 ;
17                 System.out.println ( " \n - = <  Valores do Terreno  > = - " );
18                 System.out.println ( " Largura do terreno      : " + largura + " m " );
19                 System.out.println ( " Profundidade do terreno : " + profundidade + " m " );
20                 System.out.println ( " Valor do metro quadrado : R$ " + valor_m2 );
                   System.out.println ( " Área total do terreno   : " + ( largura * profundidade ) + "
21 m2 " );
22                 System.out.println ( " Valor total do terreno  : R$ " + valor );
23         }
24
25 }
```

Fonte: Adaptação do autobaseado FRANÇA NETO, L. R.



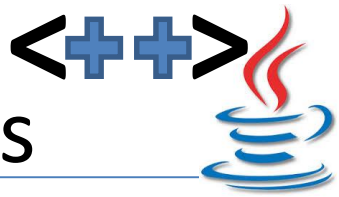
Orientação a objetos

Exemplo(s)

Versão da solução implementada de forma orientado a objetos:

Programa_OO.java

```
1  public class Programa_OO {
2
3
4          public static void main ( String [ ]  args ) {
5                  // Programa usando orientação a objeto.
6                  Terreno lote ;
7                  lote = new Terreno ( ) ;
8                  lote.entraValores ( ) ;
9                  lote.exibeValores ( ) ;
10
11         }
12
13 }
```



Exemplo(s)

Versão da solução implementada de forma orientado a objetos:

Terreno.java

```
1  import java.util.Scanner;
2
3  public class Terreno {
4
5          private Retangulo dimensao ;
6          private double valor_m2 ;
7
8
9          // construtor sem argumentos.
10         public Terreno ( ) {
11           this.dimensao = new Retangulo ( ) ;
12           this.valor_m2 = 0 ;
13         }
14
           // construtor que permite especificar a largura, a
15 profundidade e o valor do metro quadrado.
           public Terreno ( double largura , double pro-
16 fundidade , double valor_m2 ) {
                   this.dimensao = new Retangulo ( lar-
17 gura , profundidade ) ;
18                 setValor_m2 ( valor_m2 ) ;
19         }
20
21         // obtém o valor do metro quadrado.
22         public double getValor_m2() {
23                 return valor_m2;
24         }
```

Orientação a objetos

Exemplo(s)

Versão da solução implementada de forma orientado a objetos:

```
25
26          // define o valor do metro quadrado.
27          public void setValor_m2(double valor_m2) {
28                  if ( valor_m2 > 0 ) {
29                          this.valor_m2 = valor_m2;
30                  }
31                  else
32                  {
                      System.out.println ( " Valor do metro
33  quadrado a largura e/ou altura inválido ! " ) ;
34                  }
35          }
36
37            // obtém a profundidade.
38            public double getProfundidade ( ) {
39              return dimensao.getAltura ( ) ;
40            }
41
42            // define a profundidade.
              public void setProfundidade ( double profundi-
43  dade ) {
44                      dimensao.setAltura ( profundidade ) ;
45            }
46
47            // obtém a largura.
48            public double getLargura ( ) {
49              return dimensao.getLargura ( ) ;
50            }
51
52            // define a largura.
53            public void setLargura ( double largura ) {
54                      dimensao.setLargura ( largura ) ;
55            }
56
              // este método retorna a área do terreno ( em
57  metros quadrados ).
58            public double getArea ( ) {
59              return dimensao.getArea ( ) ;
60            }
61
              // este método retorna o valor do terreno em
62  função do valor do metro quadrado.
63            public double getValorTerreno ( ) {
64              return ( dimensao.getArea ( ) * valor_m2 ) ;
```