Programação Orientada a Objetos

UNIDADE 1

GUIA DE ESTUDO

UNIDADE 1

PROGRAMAÇÃO ORIENTADA A OBJETOS

PARA INÍCIO DE CONVERSA

Caro(a) aluno(a), vamos iniciar nosso curso de Programação Orientada a Objetos.

Esse é o seu primeiro guia de estudo da disciplina. Vamos falar um pouco sobre a disciplina e o que será abordado nessa primeira unidade.

Veremos um pouco da estrutura do curso de Programação Orientada a Objetos. Esse curso será dividido em quatro unidades. Para cada unidade teremos um guia de estudos para auxiliar você em cada uma dessas etapas.

As unidades terão os seguintes conteúdos:

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

- Unidade 2
    1. Conceitos básicos e terminologias de programação orientada a objetos.
    2. Classe, objetos, atributos, métodos, construtores e sobrecarga.
    3. Instanciação e referência de objetos.
    4. Envio de mensagens.
    5. Ciclo de vida de um objeto.

- Unidade 3
    1. Abstração e encapsulamento.
    2. Herança.
    3. Criação e uso de hierarquia de classes.
    4. Interfaces e classes abstratas.

- Unidade 4
    1. Relacionamento entre objetos: composição, associação, dependência e herança.
    2. Polimorfismo.
    3. Ligação dinâmica (dynamic binding).
    4. Tratamento de exceções.

DICA

É importante que você utilize esse guia como uma referência para seus estudos. Mas, o uso de outras referências como os livros indicados, será fundamental para o seu desenvolvimento nesse Curso.

Assim, como toda disciplina de programação, a prática é indispensável para o aprendizado. Você deve fazer as atividades práticas, resolver o máximo de exercícios que for capaz. Costumo dizer que, quanto mais exercícios você fizer, mais experiência vai adquirir. E que fazer exercício de programação demais não tem nenhuma contraindicação!

Portanto, lembre que praticar os conceitos apresentados aqui nesse Curso é fundamental para um bom aprendizado da disciplina de programação orientada a objetos.

REFERÊNCIAS BIBLIOGRÁFICAS

Gostaria de lhe indicar como referência os seguintes livros:

- DEITEL, H. M. Java: Como Programar. 6ª edição. São Paulo: Pearson Prentice Hall, 2005.
- CORNELL, G., HORSTMANN, C. Core Java 2 Volume I - Fundamentos. Ed. Alta Books, 2005.
- ASCENCIO, A. F. G., CAMPOS, E. A. V. Fundamentos da Programação de Computadores. 2ª Edição. São Paulo: Prentice Hall, 2008. 448p.
- GONÇALVES, E. Dominando o Eclipse. Ed. Ciência Moderna, 2006.
- SEBESTA, R.W. Conceitos de linguagens de programação. 5 ed. Porto Alegre: Bookman, 2003.

Além desses livros, você pode encontrar vários outros livros, apostilas, sites, vídeos e etc., que tratam dos assuntos abordados no curso de programação orientada a Objetos.

Preparado para um turbilhão de informações?

Espero que você goste, pois, meu intuito é de lhe auxiliar junto ao seu tutor no que for preciso.

ORIENTAÇÕES DA DISCIPLINA

Nessa unidade veremos uma introdução ao assunto, conceitos básicos e termos usados em programação orientada a objetos. Procure ler nos livros indicados e em outras referências mais um pouco sobre esses conceitos básicos que iremos apresentar aqui. É importante que você tenha todos os conceitos e termos bem fixados, pois usaremos esses termos e conceitos no decorrer do curso.

INTRODUÇÃO À ORIENTAÇÃO A OBJETOS

Caro(a) aluno(a), para darmos início a nossa disciplina, veremos um pouco sobre a tarefa de programação de computadores para começar a falar sobre orientação a objetos. Desde a invenção do computador, a tarefa de programar vem sofrendo mudanças constantes ao longo dos anos. A primeira razão para as mudanças é acomodar o aumento da complexidade dos programas.

EXEMPLO

Por exemplo, quando os computadores foram inventados, a programação era feita 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.

Com o aumento da complexidade dos programas, foi criada uma ferramenta para facilitar a geração desses códigos. Foi criada uma linguagem de montagem para auxiliar nessa tarefa, a linguagem Assembly. Depois foram criadas ferramentas para a tradução de uma linguagem mais fácil para os programadores entenderem. As linguagens mais próximas do entendimento do ser humano são chamadas de linguagens de alto nível, enquanto as mais próximas do hardware são chamadas de linguagens de baixo nível.

Essas ferramentas de tradução são conhecidas como compiladores ou tradutores. Uma dos primeiros compiladores desenvolvidos para tradução de linguagem de alto nível em linguagem de baixo nível (linguagem de máquina) foi o compilador FORTRAN (IBM Mathematical FORmula TRANslation System) da IBM em 1954. FORTRAN deu origem também a outras linguagens como o BASIC que foi bastante usada em microcomputadores das décadas de 1970 e 1980, sendo nesse caso, usados softwares tradutores no lugar de compiladores. Todas essas linguagens da época eram linguagens imperativas, assim como a linguagem Assembly, não mudando a metodologia nem o paradigma de programação.

Com a evolução das linguagens de programação, foi criada nos anos 60 a programação estruturada. Exemplos importantes desse tipo de programação são as linguagens C e Pascal. Usando-se linguagens estruturadas, foi possível, pela primeira vez, escrever programas moderadamente complexos de maneira razoavelmente fácil. Entretanto, com programação estruturada, quando um projeto atinge certo tamanho, torna-se extremamente difícil e muito custoso efetuar sua manutenção e fazer qualquer modificação. Atualmente, muitos projetos estão próximos ou no ponto em que o tratamento estruturado não mais funciona. Mesmo com modularização dos programas. Para resolver esse problema, a programação orientada a objetos foi criada.

Com a programação orientada a objetos, podemos melhorar os conceitos vistos em programação estruturada com o uso de modularização para dividir o problema em partes menores e assim, solucionar um grande problema. Com a orientação a objetos temos como dividir de forma mais independente as partes de um programa maior em objetos. Além de enxergar o problema de outra maneira, onde teremos componentes independentes que funcionam de forma completa, com pouca ou nenhuma dependência um dos outros.

A orientação a objetos não é apenas programação. Essa metodologia envolve desde a fase de análise e projeto até chegar à programação. Para esclarecer melhor o termo orientação a objetos, vamos falar um pouco sobre o processo de desenvolvimento usando essa metodologia.

Com o uso da metodologia orientada a objetos em todas as etapas do desenvolvimento, o uso de programação orientada a objetos se torna uma forma natural de programar essa solução.

PALAVRAS DO PROFESSOR

Na disciplina anterior, você aprendeu os primeiros conceitos básicos de programação e da linguagem Java. Desenvolveu seus primeiros programas simples, usando essa linguagem de programação. Tudo o que foi ensinado e tudo o que você criou foi no modelo de programação estruturada, mesmo usando uma linguagem de programação orientada a objetos. Nesse modelo de programação estruturada, você viu que fica mais fácil separar os programas em procedimentos. Ou seja, módulos ou métodos.

Esses pequenos módulos executam funções específicas, interagem com outros módulos ou com o programa principal e podem retornar ou não dados. Eles também são reutilizados. Na programação orientada a objetos veremos uma forma diferente de encarar os problemas e propor soluções diferentes das vistas até o momento.

VOCÊ SABIA?

Você sabia que a programação orientada a objetos utiliza os conceitos que aprendemos no jardim de infância? Pois é, são eles: objetos e atributos, todos e partes, classes e membros. É difícil explicar por que demoramos tanto a aplicar estes conceitos à análise e especificação de sistemas de informações - talvez porque estivéssemos ocupados demais "seguindo a boiada" durante o auge da análise estruturada para imaginar que havia alternativas. Ou talvez, porque o hardware até então não a tinha capacidade de processamento necessária para o uso de programação orientada a objetos.

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. Programação orientada a objetos, técnicas e artefatos ditos "orientados a objetos" incluem linguagens, sistemas, interfaces, ambientes de desenvolvimento, bases de dados e etc.

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.

EXEMPLO

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

Caro(a) aluno(a), durante a minha experiência como professor de programação orientada a objetos, principalmente no final da década de 90, quando ainda era muito difundida a programação estruturada, percebi que os alunos que já eram programadores experientes tinham inicialmente grande dificuldade em migrar para a orientação a objeto. Eles tinham que esquecer os anos de prática analisando os problemas de forma estruturada, para reaprender a analisá-los de forma voltada a objeto. É claro que ninguém faria isto se não tivesse bons motivos.

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.

LEITURA COMPLEMENTAR

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. Como os livros que temos na biblioteca virtual:

- 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

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.

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.

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 }
```

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         }
```

```
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 ) ;
```

```
65              }
66
                // este método entra, via teclado, os valores do
67  terreno.
68              public void entraValores ( ) {
                        Scanner leia = new Scanner ( System.
69  in ) ;
70                      double valor ;
                        System.out.print ( " Digite o valor da
71  largura do terreno: " );
72                      valor = leia.nextDouble ( ) ;
73                      dimensao.setLargura ( valor ) ;
                        System.out.print ( " Digite o valor da
74  profundidade do terreno: " );
75                      valor = leia.nextDouble ( ) ;
76                      dimensao.setAltura ( valor ) ;
                        System.out.print ( " Digite o valor do
77  metro quadrado do terreno: " );
78                      valor = leia.nextDouble ( ) ;
79                      this.setValor_m2 ( valor ) ;
80              }
81
                // este método exibe na tela de console os valo-
82  res do terreno.
83              public void exibeValores ( ) {
                        System.out.println ( " \n - = <  Valores
84  do Terreno  > = - " );
                        System.out.println ( " Largura do terre-
85  no     : " + dimensao.getLargura ( ) + " m " );
                        System.out.println ( " Profundidade do
86  terreno : " + dimensao.getAltura ( ) + " m " );
                        System.out.println ( " Valor do metro
87  quadrado : R$ " + this.getValor_m2 ( ) );
                        System.out.println ( " Área total do ter-
88  reno   : " + this.getArea ( ) + " m2 " );
                        System.out.println ( " Valor total do ter-
89  reno  : R$ " + this.getValorTerreno ( ) );
90              }
91
92
93  }
```

Retangulo.java

```
1  public class Retangulo {
2          private double largura; // largura do retângulo
3          private double altura; // altura do retângulo
4
5          // construtor sem argumentos.
6          public Retangulo ( ) {
7           this.largura = 1 ;
8           this.altura = 1 ;
9          }
10
11         // construtor que permite especificar a largura e a altura.
12         public Retangulo ( double largura , double altura ) {
13          if ( ( largura > 0 ) && ( altura > 0 ) ) {
14                  this.largura = largura ;
15          this.altura = altura ;
16          }
17          else
18          {
19                 System.out.println ( " Valor da largura e/ou altura
inválido ! " ) ;
20          }
21         }
22
23         // obtém a altura.
24         public double getAltura ( ) {
25          return altura ;
26         }
27
28         // define a altura.
29         public void setAltura ( double altura ) {
30                  if ( altura > 0 ) {
31                                  this.altura = altura ;
32                  }
33                  else
34                  {
35                          System.out.println ( " Valor da altura inváli-
do ! " ) ;
36                  }
37         }
38
39         // obtém a largura.
40         public double getLargura ( ) {
```

```
41              return largura ;
42          }
43
44          // define a largura.
45          public void setLargura ( double largura ) {
46                  if ( largura > 0 ) {
47                      this.largura = largura ;
48                  }
49                  else
50                  {
51                      System.out.println ( " Valor da largura inválido ! " ) ;
52                  }
53          }
54
55          // este método retorna a área do retângulo ( em metros quadrados ).
56          public double getArea ( ) {
57              return ( this.largura * this.altura ) ;
58          }
59
60      }
```

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

Observe, na solução estruturada, o programa basicamente é dividido em três etapas: Entrada de dados (linhas de 10 a 15); processamento (linha 16); e saída de dados (respostas, linhas de 17 a 22).

Como pode ser visto a solução orientada a objeto, parece muito mais complexa do que a versão estruturada. Isso ocorre porque a solução orientada a objetos é bem mais completa e permite ser reutilizada, em partes, em outros programas. Aparentemente o trabalho é maior, porém depois que se passa a reutilizar parte desse código, veremos que essa solução (OO) é mais interessante.

É fato também que, para problemas simples como esse, a abordagem orientada a objetos não apresenta uma grande vantagem em relação à estruturada. O que fica realmente claro é que temos outra maneira de resolver o mesmo problema.

LINGUAGENS TÍPICAS ORIENTADAS A OBJETOS

PALAVRAS DO PROFESSOR

Veremos agora um pouco da história das linguagens de programação orientada a objetos. A primeira linguagem a incorporar facilidades para definir classes de objetos genéricos na forma de uma hierarquia de classes e subclasses foi à linguagem SIMULA, que foi idealizada em 1966, na Noruega, como uma extensão da linguagem ALGOL 60.

Então, uma classe em SIMULA é um módulo englobando a definição da estrutura e do comportamento

comuns a todas as suas instâncias (objetos), veremos mais detalhes sobre esses termos adiante. Como o nome indica, é uma linguagem adequada à programação de simulações de sistemas que podem ser modelados pela interação de um grande número de objetos distintos.

As ideias de SIMULA serviram de base para as propostas de utilização de tipos abstratos de dados, e também para o desenvolvimento da linguagem de programação Smalltalk.

DICA

Assim como o mouse e as interfaces gráficas (GUI), Smalltalk foi desenvolvida no Centro de Pesquisas da Xerox durante a década de 70, e Alan Kay foi um de seus idealizadores. Além da Xerox, que criou a ParcPlace Systems especialmente para comercializar Smalltalk-80 e seus sucedâneos (objectWorks), a Digitalk lançou em 1986 uma versão de Smalltalk para ambiente DOS, e mais recentemente a versão para Windows, o que contribuiu para uma maior difusão da linguagem [Digitalk].

Caro(a) aluno(a), de fato a linguagem Smalltalk se tornou um referencial de linguagem orientada a objeto. Nessa linguagem tudo é objeto, não existem tipos primitivos. Ela é considerada como uma linguagem orientada a objetos pura. Nos anos 90 com o desenvolvimento dos PCs e suas interfaces gráficas, a linguagem Smalltalk teve uma boa aceitação no mercado e na academia. Com ela eu tive minha primeira experiência com orientação a objetos, ainda na graduação. Lembro que o desempenho ainda não era bom em virtude do hardware que não era suficiente para aplicações comerciais de grande porte. Mesmo assim, já havia alguns projetos sendo desenvolvidos no Banco do Brasil que usavam Smalltalk.

VOCÊ SABIA?

Você sabia que existia também uma confusão muito comum entre linguagens orientadas a objetos, linguagens orientadas a eventos e linguagens com interface gráfica? Pois é, essa confusão inicial se deu ao fato que de na época da popularização das linguagens orientadas a objetos, também foi à época do surgimento dos sistemas operacionais de interface gráfica e orientados a eventos. Fora isso, não tem nenhuma relação necessária das linguagens orientadas a objetos possuírem interface gráfica. Para aumentar essa confusão, Smalltalk, que era bastante representativa, usava interface gráfica.

Além de Smalltalk, outras linguagens orientadas a objetos têm sido desenvolvidas. Notadamente C++, uma extensão de C, Objetive-C, outra extensão de C, menos popular que a anterior, Pascal orientado a objetos, Eiffel e mais recentemente, JAVA e C#. Essas linguagens orientadas a objetos, assim como outras linguagens orientadas para objetos, têm sido usadas em aplicações variadas onde a ênfase está na simulação de modelos de sistemas, como automação de escritórios, animação gráfica, informática educativa, instrumentos virtuais, editores de texto e bancos de dados em geral, entre outras. Praticamente, qualquer tipo de software hoje é desenvolvido usando linguagem orientada a objetos.

PROGRAMAÇÃO ORIENTADA A OBJETO EM JAVA

A linguagem JAVA e um pouco de história: James Gosling, Mike Sheridan, e Patrick Naughton iniciou o projeto da linguagem Java em junho de 1991, na Sun Microsystems. Java foi originalmente concebido para a televisão interativa, mas era muito avançada para a indústria de televisão a cabo digital no momento. A linguagem era inicialmente chamada Oak depois de Project Green e por fim, foi renomeada para Java. Foi concebida com o estilo e sintaxe semelhantes às linguagens C / C ++, deixando os programado-

res dessas linguagens mais familiares com o Java.

Java é uma linguagem de programação orientada a objetos. Todo programa em Java usa classes e objetos, e compreender esses conceitos é fundamental para compreender a própria linguagem. Na prática, sistemas de software reais são grandes e precisam ser fatorados em partes relativamente independentes para serem viáveis. Como em Java isso é feito com classes e objetos. Os objetos são entidades distintas que são usadas para criar os programas. Os objetos são completos, comportando dentro deles tanto código como dados, ou seja, as suas funções ou operações e os seus dados que são chamados de atributos. As operações ou funções dos objetos são chamadas de métodos. Esses métodos servem para vários propósitos, pode ser usado para modificar atributos do objeto, fornecer funcionalidades para que outros objetos usem esse, etc. Essas operações são chamadas através de mensagens. Os objetos são criados ou instanciados usando a classe que os modelam. Essa classe é como o molde para o objeto. Contém as características dos atributos e os códigos dos métodos (operações).

DICA

Segundo o site da ferramenta, Java é a base para praticamente todos os tipos de aplicações em rede e é o padrão global para o desenvolvimento e distribuição de aplicações móveis e incorporadas, jogos, conteúdo baseado na Web e softwares corporativos. Conforme seus mantenedores, atualmente Java conta com mais de 9 milhões de desenvolvedores em todo o mundo, de forma eficiente, o Java permite que você desenvolva, implante e use aplicações e serviços estimulantes. De laptops a datacenters, consoles de games a supercomputadores científicos, telefones celulares a Internet, o Java está em todos os lugares!

Programação orientada a objetos pode ser feita utilizando várias linguagens de programação disponíveis atualmente. Os princípios da orientação a objetos são os mesmos, podendo ser aplicados em várias linguagens. Em nossa disciplina iremos utilizar a linguagem Java. Como foi visto anteriormente, Java tem várias vantagens e também foi utilizada na disciplina anterior de programação. Como isso, o tempo de aprendizado fica menor, pois o aluno já conhece a sintaxe e ferramentas que são usadas na linguagem Java.

Ufa! Vamos respirar agora.

PALAVRAS DO PROFESSOR

Bem caro(a) aluno(a), chegamos ao fim da unidade 1.

Nela vimos vários conceitos e termos usados em Orientação a Objetos. Qualquer dúvida entre em contato com o seu tutor.

Aguardo você na próxima unidade. Até lá!