


[< VOLTAR](#)

Introdução à Programação Orientada a Objetos com Java

Chegou a hora de começar a colocar a "mão na massa" para se aprender essa importante linguagem de programação: Java. Aqui você aprenderá os primeiros conceitos de orientação a objetos, a preparar o ambiente para programação Java SE (Standard Edition) e terminará essa aula programando em Java já com orientação a objetos.

NESTE TÓPICO

- Orientação a Objetos
- Classes, atributos e métodos em Java
- Para ficar um pouco mais claro: Vamos programar 
- Resumo dessa aula
- Referências



Orientação a Objetos



Em programação Java o conceito de orientação a objetos (OO) é intrínseco e importantíssimo. Por conta disso, conhecer muito bem o conceito de objeto é muito importante para este novo paradigma de programação, antes de sair programando.

Para este importante conceito, será utilizada uma breve menção com o primeiro filme Matrix, de 1999. Fica aqui, uma **sugestão** para que você assista (ou reveja) este filme, para melhor entendimento de alguns conceitos.

No filme, o personagem principal, Neo, é um bom programador em uma empresa conceituada e tem uma vida comum, até surgir um homem misterioso, Morpheu, lhe oferecendo duas pílulas: Uma azul e uma vermelha. Se Neo escolhesse a pílula azul, ele se libertaria e conheceria a verdade, conhecendo a Matrix; se escolhesse a pílula vermelha, ele voltaria a sua rotina normal e esqueceria daquilo. Neo escolhe a pílula azul e desmaia logo em seguida. Quando acorda ele está sendo retirado de uma câmara cheia de fluido e diversos fios conectados a seu corpo começam a se soltar. Ele então chega numa sala cheia de computadores, onde encontra

novamente Morpheu e sua equipe. Morpheu explica a ele que tudo que ele tinha vivido até aquele momento (que ele chamava de realidade) era gerado por um programa de computador e que, naquele momento, ele está na realidade, próximo às máquinas que geravam aquela realidade que ele vivia, contudo, essas máquinas são inimigas.



Matrix



Imagine agora que você está na Matrix, ou seja, que isso que você está vivendo agora é uma realidade gerada por um computador. Agora perceba tudo que está a sua volta: O computador, o celular, papéis, canetas etc. Tudo que está a sua volta é um **objeto**, certo?

Se você está inserido em um programa de computador, então existe uma programação (**código**) gerando esse software (chamado vida real). Os **objetos** que estão a sua volta são frutos dos trechos de código que os geram. Por exemplo, existem milhares de objetos tipo celular no mundo, mas é necessário apenas um trecho de código para gerar um celular.

Este é o conceito de **classe** e **objeto**. A classe é o trecho de código onde nós programamos. São as classes que dão origem aos objetos, **que só existem no programa em execução**. Em outras palavras nós criamos classes que, quando nosso programa estiver sendo executado, geram os objetos. Podemos dizer ainda que a classe é a ?receita de bolo? do objeto. Uma classe pode gerar vários objetos com características e comportamentos diferentes.

Por exemplo, você é um ser humano, certo? Você possui suas próprias características físicas, como cor dos olhos, cor dos cabelos, altura, peso etc. Nenhum ser humano no mundo é 100% igual a você (mesmo se você for gêmeo). Você é único. De qualquer forma podemos dizer que você pertence a uma classe chamada **Pessoa**, afinal de contas, você é uma pessoa. Todas as pessoas do mundo (lembre-se: que são diferentes umas das outras) pertencem a única e mesma classe: **Pessoa**.

Dentro dessa classe **Pessoa** há locais para se colocar as características e comportamentos. Chamamos as características, no mundo orientado a objetos, de **atributos** e os comportamentos de **métodos**.

Continuando o exemplo: A classe **Pessoa** pode ter, por exemplo, os seguintes atributos e seus tipos (que variam de pessoa para pessoa - atenção aqui. Variam é uma palavra chave para nós, pois, em java, atributos também podem ser chamados de variáveis):

- Nome (texto)
- Cor dos olhos (texto)
- Cor do cabelo (texto)
- Sexo (caracter - uma letra)
- Altura (número real)
- Peso (número real)

Sim, sabemos que uma pessoa tem muitos outros atributos possíveis, mas vamos nos ater a estes para facilitar, por enquanto.

Reforçando: A **classe é Pessoa**; os **atributos** são cor do cabelo, cor dos olhos etc. E quais são os objetos? São as pessoas que podem ser criadas com variações dessas características. Por exemplo:

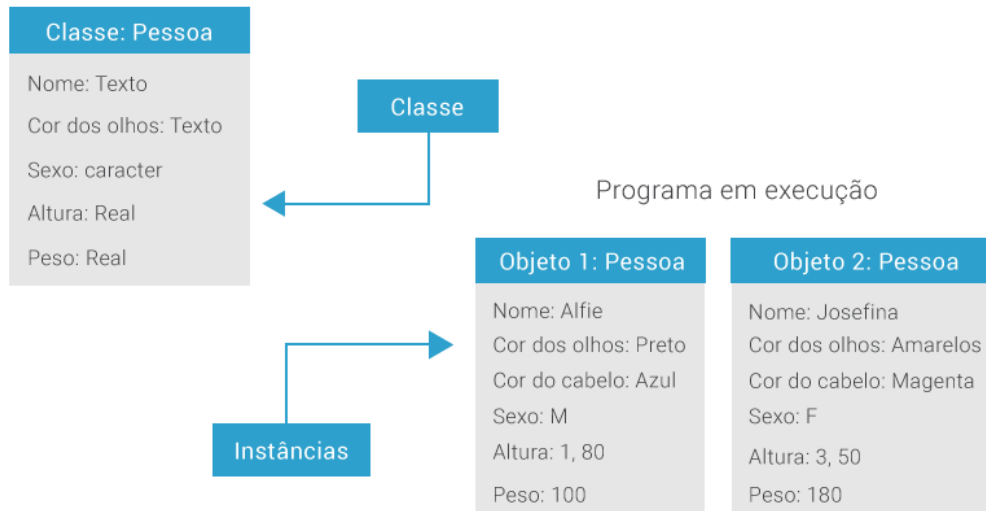


- Objeto 1: Pessoa1
 - Nome: Alfie
 - Cor dos olhos: Pretos
 - Cor do cabelo: Azul
 - Sexo: M
 - Altura: 1,80
 - Peso: 100
- Objeto 2: Pessoa2
 - Nome: Josefina
 - Cor dos olhos: Amarelos
 - Cor do cabelo: Magenta
 - Sexo: F
 - Altura: 3,50
 - Peso: 180

Independente de quão estranho possam parecer estes exemplos, é importante entender que cada uma dessas pessoas criadas é **um objeto** da classe **Pessoa**?. No mundo orientado a objetos, dizemos que cada objeto é uma **instância** da classe que o cria. A palavra **instância** é chave para nós.

CLASSE X INSTÂNCIA

Ambiente de desenvolvimento



Os objetos são instâncias da classe



Essa é a grande “sacada” da orientação a objetos: Uma classe que pode criar vários objetos, como uma “fábrica de objetos”, mas a coisa não para aí: Um objeto pode se relacionar com outro(s) ou ser criado por outro(s). Estes conceitos serão melhor compreendidos quando estivermos programando e exemplificando.

Podemos ter, por exemplo, uma classe chamada Endereço e poderemos associar a pessoa a um endereço. Poderemos fazer, ainda, com que um endereço consiga criar um bairro ainda não cadastrado. Tudo é possível no maravilhoso mundo orientado a objetos. Perceba que a classe Endereço, não precisa se relacionar apenas com uma pessoa, pode relacionar-se com uma empresa e se for preciso dar manutenção no endereço, apenas uma classe é afetada. Uma das grandes vantagens da orientação a objetos é essa: **Casa coisa deve ficar em seu devido lugar.**

Em resumo, as **classes** são “moldes” que programamos e que geram os objetos em tempo de execução, ou seja, quando o programa está rodando. Esses **objetos** são chamados de **instâncias** das classes que codificamos. O que define as **características** dos **objetos** são seus **atributos** e os **comportamentos**, seus **métodos**.

Segundo Teruel, programar de forma estruturada (como foi feito até o momento) é mais fácil. E é mesmo, pois orientação a objetos é um paradigma totalmente novo; uma nova forma de pensar. Contudo, imagine que você está desenvolvendo uma aplicação para uma grande corporação, onde é preciso manipular dados financeiros, de clientes, funcionários, gestão

etc. Separar as coisas em seus devidos lugares lhe facilitará muito na hora de realizar alterações ou dar manutenção em seu código. Acredite, o mundo orientado a objetos é muito mais fácil de trabalhar que o mundo procedimental (paradigma antigo), pois cada coisa estará em seu lugar, mas agora, é hora de programar.

Classes, atributos e métodos em Java

Este é um dos tópicos mais importantes desta disciplina, pois o entendimento destes conceitos é fundamental para programar corretamente em Java, utilizando orientação a objetos.

Para programarmos em Java, é muito importante seguirmos as boas práticas de programação que definem a estrutura de uma classe, que deve ser composta, nessa ordem, por:

1. Importações de outras classes e pacotes (vistos em breve)
2. Definição do pacote o qual a classe pertence
3. Definição da classe
4. Atributos locais
5. Métodos

Os **atributos** são as **variáveis locais** que pertencem a classe e são por ela utilizados. Os atributos podem ser de vários tipos: texto, caractere, número inteiro, número real etc. Haverá um tópico específico para falarmos sobre esses diferentes tipos de atributos.

Os **métodos** são, conforme mencionado anteriormente, os **comportamentos** das classes, ou seja, são "funções" que podem ser chamadas para executar alguma tarefa.

Os métodos

Conforme mencionado, os métodos possuem suas particularidades e podem ser de vários tipos.

Um método, nada mais é que a execução de alguma tarefa quando necessário. Para essa execução, os métodos podem receber informações (dados) externas e podem retornar valores para quem o chamou. Sim, os métodos precisam ser chamados.

Se um método retorna um valor, dizemos que o método é "**tipado**", ou seja, ele conterà uma informação de algum tipo que pode ser um texto, um número etc. Se o método não retorna nada para quem o chamou, então ele é "vazio", ou, como dizemos em Java, "**void**".



Veja um exemplo de método vazio que apenas imprime, em console, o nome do autor:

```
1.    //...
2.    public void imprimeNomeAutor(){
3.        System.out.println("O autor deste código é o sr Josefino José");
4.    }
5.    //...
```

Veja, agora, um exemplo de um método tipado, que recebe dois valores e retorna a soma deles:

```
1.    //...
2.    public int soma(int a, int b){
3.        return a + b;
4.    }
5.    //...
```

Note que no método de soma dois valores são necessários para sua execução. Neste caso, eles são passados via “parâmetro” dos métodos, ou seja, quem invocar este método deverá passar, também, os valores que devem ser somados. Estes conceitos ficarão mais claros quando tudo isso for utilizado junto.

Classes



Uma classe em Java pode ser instanciada por outra, ou seja, quando seu programa estiver sendo executado, um objeto poderá se relacionar a outro, através de suas instruções.

Em Java, para uma classe referenciar outra, utiliza-se a palavra reservada “**new**”. Ao utilizar a instanciação de uma classe, o **método construtor** da classe é chamado automaticamente.

Por exemplo, imagine a classe “Pessoa” criando um endereço. Sua estrutura será assim:

```
1.    public class Pessoa(){
2.        Endereco end = new Endereco(); //Criou-se, aqui, um atributo "end", do TIPO en
    dereço.
3.        //...
4.    }
5.    }
```

Note que a variável “end” é do TIPO endereço e, portanto, assume todas as características e comportamentos que a classe “Endereco” pode ter.

Podemos, também, invocar um construtor com passagem de parâmetros, como boas práticas de programação, contudo, neste caso, precisamos implementar o construtor na classe. O exemplo abaixo (Matematica) mostra

o construtor implementado manualmente e a passagem de parâmetros para ele pela classe "Principal".

Para ficar um pouco mais claro: Vamos programar

Para exemplificar como isso funciona em Java, vamos implementar uma simples calculadora de números inteiros, com dois atributos locais (para armazenamento de valores a serem operados) e 4 métodos com retorno: Soma, Subtração, Multiplicação e Divisão.

Serão necessárias duas classes, isso mesmo duas! Vamos programar já utilizando o conceito de orientação a objetos com passagem de valores para métodos.

- A classe "Principal" que será responsável por criar a instância da classe responsável pelas operações matemáticas. Nessa classe, faremos a leitura dos valores via teclado.
- A classe "Matematica" que será responsável por realização das operações aritméticas

Lembre-se do conceito de orientação a objetos: Cada coisa deve ficar em seu respectivo lugar, por isso, quem executar operações aritméticas, é a classe responsável por isso que, neste caso, chama-se "Matematica".

No Netbeans, crie um novo projeto e um pacote de código fonte. Crie a classe "Matematica" primeiro:



```
1. public class Matematica {
2.
3.     //Atributos locais
4.     int a, b;
5.
6.     //Construtor da classe, que recebe dois valores (X e Y) e atribui
7.     //aos valores locais (a e b);
8.     public Matematica (int x, int y){
9.         a = x;
10.        b = y;
11.    }
12.
13.    //Metodo de soma
14.    public int soma(){
15.        return a + b;
16.    }
17.
18.    //Metodo de subtracao
19.    public int subtrai(){
20.        return a - b;
21.    }
22.
23.    //Metodo de multiplicacao
24.    public int multiplica(){
25.        return a * b;
26.    }
27.
28.    //Metodo de divisao
29.    public int divide(){
30.        return a / b;
31.    }
32. }
```

Depois crie a classe principal, que deverá ser assim:




```

1. import java.util.Scanner;
2.
3. public class Principal {
4.
5.     public static void main(String args[]) {
6.
7.         //Define o leitor do teclado
8.         Scanner leitor = new Scanner(System.in);
9.
10.        //Define a variáveis locais
11.        int x, y;
12.
13.        //Le:
14.        System.out.print("Informe o 1º valor: ");
15.        x = leitor.nextInt();
16.
17.        System.out.print("Informe o 2º valor: ");
18.        y = leitor.nextInt();
19.
20.        //Cria a instancia da classe matematica utilizando o construtor
21.        Matematica mat = new Matematica(x, y);
22.
23.        // Imprime o resultado das operações através
24.        // de chamadas aos métodos da classe Matemática
25.        System.out.println("-----"); //Apenas para organizar a saída
26.        System.out.println("O valor da soma é: " + mat.soma());
27.        System.out.println("O valor da subtração é: " + mat.subtrai());
28.        System.out.println("O valor da multiplicação é: " + mat.multiplica());
29.        System.out.println("O valor da divisão é: " + mat.divide());
30.        System.out.println("-----"); //Apenas para organizar a saída
31.
32.    }
33. }

```



Note que na classe principal existe uma variável local chamada “mat” que é do tipo “Matematica”, ou seja, essa variável “mat” possui todas as características e métodos da classe “Matemática”.

A saída esperada para este programa, com exemplo de valores é:

```

run:
Informe o 1º valor: 200
Informe o 2º valor: 50
-----
O valor da soma é: 250
O valor da subtração é: 150
O valor da multiplicação é: 10000
O valor da divisão é: 4
-----
CONSTRUÍDO COM SUCESSO (tempo total: 7 segundos)

```

Execução do projeto da calculadora

Lembre-se que estamos usando apenas valores inteiros aqui. Se o resultado da divisão for decimal, apenas a parte inteira será exibida.

Resumo dessa aula

Nessa aula você aprendeu sobre o importantíssimo conceito de **orientação a objetos**. Nunca se esqueça de:

- Java é orientado a objetos
- A orientação a objetos é um paradigma de pensamento e deve ser usado no desenvolvimento das aplicações
- As classes são as “receitas” dos objetos que só existem em tempo de execução. As classes são onde nós programamos
- O conceito de instância, ou seja, uma instância é um objeto de uma classe em tempo de execução
- O conceito de métodos

Pronto. Agora está pronto para programar coisas mais avançadas em Java. Tenha certeza que você entendeu o importante conceito de orientação a objetos. Não deixe de criar outros programas para praticar. Programar bem é resultado de muito estudo e muita prática. Bons estudos e boa programação.

Quiz

Exercício Final



Introdução à Programação Orientada a Objetos com Java

INICIAR ➤

Referências

Deitei P. e Deitel H., 2010, Java : Como programar, 8ª Edição, Pearson Pretice Hall

Teruel, E. C., 2015, Programação Orientada a Objetos com Java - sem mistérios - 1ª Ed., Editora Uninove

Oracle, "Hello World!" for the NetBeans IDE, disponível em <https://docs.oracle.com/javase/tutorial/getStarted/cupojava/netbeans.html>, acessado dia 10 de Março de 2016

The Matrix, 1999, IMDb, disponível em <http://www.imdb.com/title/tt0133093/>, acessado dia 18 de Março de 2016



Avalie este tópico



ANTERIOR

Introdução ao Java



Índice

Biblioteca

(<https://www.uninove.br/conhec-a->

a-

[uninove/biblioteca/sobre-](https://www.uninove.br/conhec-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)

a-

[biblioteca/apresentacao/](https://www.uninove.br/conhec-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/))

Portal Uninove

(<http://www.uninove.br>)

Mapa do Site

Ajuda?

PRÓXIMO
(<https://ava.uninove.br/seu/AVA/topico/topico.php?idCurso=>)

Tipos de dados em Curso

© Todos os direitos reservados

