Índice

[Programação Orientada a Objetos 1](#_rpzscmtcnfqn)

[Paradigmas da programação 1](#_fhmoxk71z2gw)

[Classes e objetos 3](#_o9ta5o7rgtzq)

[Compilação e Bytecode 3](#_fabw85mcxb0v)

[Modificadores de Acesso: 4](#_bko2prseptko)

[Construtores 4](#_fityvxyovizc)

[Comandos 5](#_7za0fie7rm35)

[Encapsulamento 5](#_ip72xtxyisfv)

[Métodos Get e Set 6](#_selm9a7ds2kh)

[Comandos 6](#_3pfcs84dv6ur)

[Comandos gerais 6](#_rcqot5zdvh2e)

[Métodos 6](#_t3z39jhbtd5w)

[Links e afins 7](#_v49ndfn8hwjl)

[Dúvidas 7](#_huow9eiz5of6)

[Temporários 7](#_9kkt2yc5h4iu)

# 

**Assuntos: JAVA P.O.O (Programação orientada a objetos)**

**Abertura Alunos:**

Reflexão diária: Aceitação da Mudança

Rafael Carmo e Matheus Libanio

Vídeo: Mude a sua vida – Um passinho de cada vez (aba de links)

# Programação Orientada a Objetos

## Paradigmas da programação

Paradigma de programação é o processo de pensamentos que se usa para programar, ele vai determinar quais os processos a serem usados para se programar.

**Eles são:** **orientação a objetos, programação imperativa, programação declarativa, programação orientada a eventos, programação estruturada, programação funcional, programação concorrente e paralela.**

1. **Programação Imperativa:** Centrada em alterar o estado do programa por meio de instruções. Usa estruturas como loops e variáveis para controlar o fluxo do programa.

Exemplos incluem linguagens como C e Python.

1. **Programação Declarativa:** Concentra-se mais no que deve ser alcançado, em vez de como alcançá-lo. Dois subtipos comuns são:
   1. **Funcional:** Baseado em funções puras, sem estado compartilhado, enfatizando a imutabilidade dos dados. Exemplos incluem Haskell e Lisp.
   2. **Lógica:** Define relações lógicas e deixa a máquina inferir a solução. Prolog é um exemplo.
2. **Programação Orientada a Objetos (POO):** Organiza o código em "objetos" que contêm dados e comportamentos. Usa conceitos como encapsulamento, herança e polimorfismo.

Exemplos incluem Java e C++.

1. **Programação Orientada a Eventos:** Baseada em eventos disparados por ações externas. O código responde a esses eventos. É comum em interfaces gráficas e aplicações web.

Exemplos incluem JavaScript e C#.

1. **Programação Estruturada:** Foca em dividir o programa em estruturas pequenas e gerenciáveis, como sequências, seleções e iterações. Busca clareza e facilidade de manutenção.

Exemplos incluem Fortran e Pascal.

1. **Programação Funcional:** Se concentra no uso de funções puras e evita mutabilidade e estados compartilhados. Prioriza a imutabilidade dos dados.

Exemplos incluem Haskell e Scala.

1. **Programação Concorrente e Paralela:** Lidam com a execução de múltiplas tarefas simultaneamente. Concorrente refere-se à execução de tarefas independentes em um ambiente compartilhado (ex: Python com módulo concurrent.futures), enquanto paralela refere-se à execução simultânea real (ex: usando threads ou processos).

Esses paradigmas não são mutuamente exclusivos e muitas linguagens de programação podem incorporar múltiplos paradigmas. Programadores frequentemente combinam conceitos de diferentes paradigmas para resolver problemas de forma mais eficaz.

## Classes e objetos

Classes possuem sempre nome, atributos e métodos.

* **Atributos** são características da classe, coisas que fazem parte das informações contidas na classe em si. Atributos sempre devem acompanhar com o tipo de dado (String, int, float, etc)
* **Métodos** são ações que podem ser tomadas pela classe, como verbos mesmo. Métodos sempre devem vir acompanhados de seu tipo de dado de saída, e podem ser acompanhados do tipo de dado de entrada.

O objeto não precisa carregar todas as características da classe.

O objeto também é capaz de manipular os métodos da classe.

Objetos são instâncias das classes.

**Classe e Objeto:** Uma metáfora boa pra entender Classe e Objeto é que a Classe é como se fosse a planta de um projeto (planta tipo de arquitetura, com os desenhos de como vai ser o prédio etc) e o Objeto instanciado seria o prédio já pronto utilizando os atributos definidos na planta (Classe).

Você é capaz de citar esse ‘prédio’ em diferentes partes do seu projeto, e o prédio permanece o que ele é, com tudo o que ele tem dentro.

**Palavra-chave new**

Utilizada para criar uma nova instância (objeto) de uma classe.

## Compilação e Bytecode

**Instanciar** é o ato de criar um objeto a partir de uma classe.

**Compilar** é o ato de rodar o programa. Ao ser executado ele se compila. Ele é transformado de linguagem de programação (Java, c++, python), para linguagem de computador (binário 01000111).

## Modificadores de Acesso:

* **public**, **private**, **protected**: Controlam a visibilidade dos membros da classe.
* **final:** Indica que um elemento não pode ser alterado após a criação.
* **abstract**: Define uma classe ou método como um modelo que deve ser estendido ou implementado por outras classes.

| **Modificador** | **Classe** | **Pacote** | **Subclasse** | **Mundo** |
| --- | --- | --- | --- | --- |
| **public** | ✔ | ✔ | ✔ | ✔ |
| **protected** | ✔ | ✔ | ✔ | ✘ |
| **default** | ✔ | ✔ | ✘ | ✘ |
| **private** | ✔ | ✘ | ✘ | ✘ |

## Construtores

Construtor é um método especial, definido para cada classe.

Um construtor é um método especial em uma classe que é chamado automaticamente quando um objeto é criado.

Toda classe tem pelo menos um construtor definido.

O nome do construtor é o mesmo que o nome da classe.

Ele não tem um tipo de retorno explícito (nem mesmo void).

Os construtores são usados para inicializar os atributos de um objeto e realizar qualquer outra inicialização necessária.

Ele é invocado/instanciado com o new

Se nenhum construtor for explicitamente definido pelo programador, ele cria por si só um construtor default. Se pelo menos um construtor for criado, nenhum default será gerado.

Usando diferentes listas de parâmetros, mais de um construtor pode ser definido para haver mais de uma maneira de inicializar um objeto.

### Comandos

**this.nomeAtributo = nomeAtributo;**

**This** se refere ao SEU atributo, da sua classe. O outro é o argumento, sendo argumento algo que se recebe de outra classe.

**botão direito > source > generate constructor using fields**

Comando para se gerar um construtor

**alt > shift > s** > **generate constructor using fields**

Comando alternativo para criação de construtor no STS.

**super( )** Define que o construtor pode ser uma superclasse. Superclasse é a classe da qual outras classes, chamadas subclasses, herdam atributos e métodos.

**extends** é usada para estabelecer essa relação de herança entre a superclasse e a subclasse.

## Encapsulamento

Encapsulamento em Java é um conceito que envolve o ocultamento dos detalhes internos de uma classe, protegendo seus atributos e métodos para controlar o acesso a eles por meio de modificadores de acesso, como public, private e protected.

Resumidamente, o encapsulamento permite que os dados de uma classe sejam acessados e modificados apenas por métodos específicos da própria classe.

Isso promove a segurança e a consistência dos dados, impedindo que sejam alterados diretamente de fora da classe. Além disso, possibilita a implementação de lógica adicional (como validações) ao acessar ou modificar os atributos da classe.

A prática comum de encapsulamento envolve a declaração de variáveis de instância como privadas e a disponibilização de métodos públicos (conhecidos como getters e setters) para acessar e modificar esses atributos de maneira controlada, respeitando as regras definidas pela classe. Isso ajuda a manter a integridade dos dados e facilita a manutenção do código ao encapsular o comportamento interno da classe.

### Métodos Get e Set

**Get** método de acesso. Ele busca uma informação e a retorna.

**Set** método modificador. Ele estabelece uma informação.

Com o Get e Set, o usuário passa por um processo de pedir a informação (get) e pelo processo de setar uma informação (set).

Com o get e set o usuário pede ou passa dados, e a gente pode verificar se ele tem as permissões necessárias para isso.

#### Comandos

**botão direito > source > generate getters and setters**

Comando para se gerar get e set.

**alt > shift > s** > **generate**

Comando alternativo para se gerar get e set.

**objeto.setNomeAtributo**  Comando para alterar o valor de um atributo na main.

## Comandos gerais

**[modificadorDeAcesso] class nomeDaClasse { }** Comando padrão de nomeação de classe.

Por padrão ele define a classe como public.

**NumberFormat *nome* = NumberFormat.getCurrencyInstance( );**

Comando que formata um valor de acordo com a moeda local usando a localização do sistema. Coloca R$, $, etc.

**nome.setMinimumFractionDigits (número)**

Ele determina o número máximo de dígitos fracionários.

**nome.format( Variavel)**

Comando que de fato formata a moeda

# [Métodos](https://docs.google.com/document/u/0/d/1EZSR9YwP1Rf9Nr2Fuz67siLYy3w1Xue15YKJ0xcHdtk/edit)

Clique para ser direcionado as anotações de método.

# 

# Links e afins

**Cookbook MD10, P.O.O, Programação orientada a objetos:**

<https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/01_java/10.md>

**Vídeo abertura, Mude sua vida - um passinho de cada vez:**

<https://www.youtube.com/watch?v=O_v-tIndr1M>

**Curso em vídeo, auxilio para lógica P.O.O:**

<https://www.cursoemvideo.com>

**Cookbook PR03, Projeto Conta Bancária, Modelo conta bancária:**

<https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/01_java/pr03.md>

**Apresentação PPT Luis:**

<https://docs.google.com/presentation/d/1f7lG0_iFAJvfDdw8LSnS3iHvjYeElYIY/edit#slide=id.p48>

**Número Larissa BSM:**

11 953803006

# 

# Dúvidas

# Temporários











