Sumario



[Construtores 3](#_Toc130752476)

[Tipos de Construtores 3](#_Toc130752477)

[Construtor Parametrizado 3](#_Toc130752478)

[O Construtor de Cópia 4](#_Toc130752479)

[Encapsulamento 5](#_Toc130752480)

[Getters 6](#_Toc130752481)

[Setters 6](#_Toc130752482)

[A Palavra-Chave this 6](#_Toc130752483)

# Construtores

Em Java, um construtor é um método especial que é chamado automaticamente quando um objeto é criado a partir de uma classe. Ele é usado para inicializar os atributos de um objeto e garantir que o objeto esteja em um estado válido quando é criado.

O construtor tem o mesmo nome da classe e não possui um tipo de retorno. Ele pode ter parâmetros ou não. Quando um objeto é criado, o construtor é chamado automaticamente e pode executar código que inicializa os valores dos atributos do objeto.

public class Pessoa {

private String nome;

private int idade;

public Pessoa(String nome, int idade) {

this.nome = nome;

this.idade = idade;

}

}

Neste caso, temos um construtor que recebe dois parâmetros (nome e idade) e que atribui esses valores aos atributos correspondentes da classe Pessoa.

Ao criar um objeto da classe Pessoa, o construtor é chamado automaticamente e os valores passados como parâmetros são utilizados para inicializar os atributos nome e idade.

Exemplo:

Pessoa pessoa1 = new Pessoa("João", 30);

Neste caso, um objeto pessoa1 é criado a partir da classe Pessoa, e o construtor é chamado automaticamente com os parâmetros "João" e 30.

O nome do construtor será sempre o nome da classe.

## Tipos de Construtores

### Construtor Parametrizado

Um construtor parametrizado é um tipo de construtor que recebe parâmetros. Ele é usado para inicializar os atributos de um objeto com valores específicos fornecidos pelo usuário ou por outros objetos.

Ao contrário do construtor padrão, que não recebe nenhum parâmetro, o construtor parametrizado tem um ou mais parâmetros que são usados para inicializar os atributos da classe.

Exemplo:

public class Livro {

private String titulo;

private String autor;

private int anoPublicacao;

public Livro(String titulo, String autor, int anoPublicacao) {

this.titulo = titulo;

this.autor = autor;

this.anoPublicacao = anoPublicacao;

}

}

Neste exemplo, o construtor parametrizado recebe três parâmetros: titulo, autor e anoPublicacao. Esses parâmetros são usados para inicializar os atributos correspondentes da classe Livro.

Ao criar um objeto da classe Livro usando o construtor parametrizado, precisamos passar os valores dos parâmetros. Por exemplo:

Livro livro1 = new Livro("O Senhor dos Anéis", "J.R.R. Tolkien", 1954);

Neste caso, estamos criando um objeto livro1 da classe Livro e passando os valores "O Senhor dos Anéis", "J.R.R. Tolkien" e 1954 como parâmetros para o construtor parametrizado.

O construtor parametrizado é útil quando precisamos criar objetos com valores específicos fornecidos pelo usuário ou por outros objetos. Ele nos permite criar objetos de uma forma mais flexível e personalizada, em vez de depender apenas dos valores padrão.

### O Construtor de Cópia

O construtor de cópia em é um construtor especial que permite criar uma cópia de um objeto já existente. Ele é uma forma de criar um novo objeto com os mesmos valores dos atributos de um objeto já existente.

O construtor de cópia recebe como parâmetro um objeto do mesmo tipo da classe em questão e, em seguida, cria um novo objeto com os mesmos valores dos atributos do objeto recebido como parâmetro. Isso é útil quando precisamos criar uma cópia de um objeto existente, em vez de criar um novo objeto com valores padrão.

A sintaxe para criar um construtor de cópia é semelhante à sintaxe de um construtor padrão. No entanto, em vez de inicializar os valores dos atributos com valores padrão, o construtor de cópia recebe um objeto do mesmo tipo e inicializa os valores dos atributos com os valores do objeto recebido como parâmetro.

Exemplo:

public class Pessoa {

private String nome;

private int idade;

public Pessoa(String nome, int idade) {

this.nome = nome;

this.idade = idade;

}

public Pessoa(Pessoa outraPessoa) {

this.nome = outraPessoa.nome;

this.idade = outraPessoa.idade;

}

}

Neste exemplo, temos dois construtores: um construtor padrão que recebe dois parâmetros (nome e idade) e um construtor de cópia que recebe um objeto Pessoa como parâmetro.

O construtor de cópia inicializa os valores dos atributos do novo objeto com os valores do objeto Pessoa recebido como parâmetro. Isso é feito através do uso do operador "this" para referenciar os atributos do novo objeto e do objeto Pessoa recebido como parâmetro.

Ao criar um novo objeto Pessoa usando o construtor de cópia, é necessário passar um objeto Pessoa como parâmetro. Por exemplo:

Pessoa pessoa1 = new Pessoa("João", 30);

Pessoa pessoa2 = new Pessoa(pessoa1);

Neste caso, estamos criando um objeto pessoa1 da classe Pessoa com os valores "João" e 30 como parâmetros. Em seguida, estamos criando um novo objeto pessoa2 da classe Pessoa usando o construtor de cópia e passando o objeto pessoa1 como parâmetro. O objeto pessoa2 terá os mesmos valores dos atributos do objeto pessoa1.

# Encapsulamento

Encapsulamento é o princípio de ocultar os detalhes internos de uma classe e fornecer uma interface pública para interagir com essa classe. Isso significa que os atributos e métodos de uma classe são definidos com diferentes níveis de acesso (public, private, protected e default) para controlar como os objetos daquela classe são acessados e manipulados por outras partes do programa.

A ideia por trás do encapsulamento é garantir que o estado interno de um objeto não seja acessado diretamente por outros objetos do programa. Em vez disso, o acesso aos atributos de um objeto deve ser feito através de métodos públicos da classe, que são conhecidos como getters e setters.

Os getters são métodos públicos que permitem obter o valor de um atributo de uma classe. Já os setters são métodos públicos que permitem definir o valor de um atributo de uma classe. Ambos os métodos são importantes para garantir que o acesso aos atributos de uma classe seja controlado e seguro.

Exemplo

public class Pessoa {

private int idade;

public int getIdade() {

return idade;

}

public void setIdade(int idade) {

this.idade = idade;

}

}

Neste exemplo, o atributo idade é definido como privado, o que significa que ele só pode ser acessado dentro da classe Pessoa. Os métodos getIdade() e setIdade() são definidos como públicos para permitir o acesso ao atributo idade de fora da classe.

Ao definir o atributo idade como privado e fornecer getters e setters públicos, estamos encapsulando o estado interno da classe Pessoa. Isso significa que os objetos da classe Pessoa podem ser usados sem que outros objetos do programa saibam ou modifiquem diretamente o valor da idade, garantindo assim um controle mais seguro e estruturado sobre a manipulação dos dados.

## Getters

Um getter é um método público que permite obter o valor de um atributo de uma classe. Ele não recebe parâmetros e retorna o valor atual do atributo

## Setters

Já o setter é um método público que permite definir ou modificar o valor de um atributo de uma classe. Ele recebe um parâmetro que representa o novo valor a ser definido para o atributo e, em seguida, atualiza o valor do atributo interno da classe.

## A Palavra-Chave this

A palavra-chave "this" é usada para referenciar o objeto atual em que o código está sendo executado. Isso é útil quando há uma ambiguidade entre variáveis de instância de uma classe e variáveis locais em um método, por exemplo.

Quando um método ou construtor é chamado em um objeto, o "this" refere-se a esse objeto.

Exemplo:

public class Ex {

public static void main(String[] args) {

Cachorro obj = new Cachorro();

Obj.setNome(“Marley”);

System.out.println(obj.getNome());

}

}