

Construtor

O que é um construtor?

- É um método especial de uma classe que é automaticamente chamado quando um objeto dessa classe é criado.
- Tem a responsabilidade de inicializar os atributos do objeto, ou seja, atribuir valores iniciais a eles.
- O construtor pode receber parâmetros para personalizar a criação do objeto, e geralmente não retorna nenhum valor.
- Tem o mesmo nome da classe e é utilizado para garantir que o objeto seja corretamente configurado no momento de sua criação.
- É uma boa prática fazer validações no construtor para garantir que o objeto é criado num estado válido.
- É chamado automaticamente quando usamos `new NomeDaClasse()`.

Se não definirmos um construtor, o Java cria um **construtor padrão** sem parâmetros automaticamente.

O que realmente cria o objeto na memória é o `new`, mas o **construtor define como o objeto é inicializado**.

💡 **Processo de criação de um objeto:**

- 1 O `new` reserva espaço na memória para o objeto.
- 2 O construtor é chamado automaticamente para inicializar os atributos.
- 3 O objeto é devolvido pronto para ser usado.

Quando fazer validações dentro do construtor?

- ✓ Se os atributos forem obrigatórios (ex.: um `Cliente` deve ter um `nome`).
- ✓ Se um valor for nulo pode causar erros futuros.
- ✓ Se queremos garantir que o objeto não fica num estado inválido logo na criação.

Quando Criar um Construtor Padrão (Vazio)?

- ✓ Se quisermos permitir criar um objeto **sem definir os atributos imediatamente**, podemos usar um **construtor vazio** e depois configurar os atributos com setters.
- ✓ Permitir a criação do objeto antes de definir valores faz sentido no design.

Razões para NÃO Criar um Construtor Vazio Desnecessário

📌 1. Evita Criar Objetos em Estados Inválidos

💡 Se uma classe exige atributos obrigatórios, um construtor vazio permite criar um objeto sem esses dados, o que pode levar a erros.

📌 2. Reduz Ambiguidade e Evita Má Prática

💡 Se uma classe tem um construtor sem argumentos e outro com argumentos, pode ser confuso saber qual usar.

📌 3. Segue o Princípio "Tell, Don't Ask"

💡 O objeto deve ser criado já com os dados corretos, em vez de ser criado vazio e preenchido depois.

Quando Criar Múltiplos Construtores? (Sobrecarga)

💡 Usamos sobrecarga de construtores para flexibilidade:

- ✓ Se houver **diferentes formas de inicializar um objeto**.
- ✓ Se alguns atributos forem **opcionais**.
- ✓ A melhor abordagem é criar:
 - 1 Um construtor com os atributos obrigatórios.
 - 2 Outro construtor que inclua também os opcionais (sobrecarga).

Se houver **muitos atributos opcionais**, um **Builder Pattern** pode ser uma melhor alternativa:

(Builder Pattern é um padrão de design criacional que permite construir objetos complexos passo a passo, evitando múltiplos construtores e melhorando a legibilidade do código. Em vez de passar todos os parâmetros no construtor de uma só vez, o Builder permite configurar o objeto de forma flexível antes de criá-lo).

1 O Que É Um Construtor?

 Pergunta possível:

 O que é um construtor em Java e qual a sua finalidade?

 Resposta:

Um **construtor** é um **método especial** que tem o **mesmo nome da classe** e é chamado automaticamente quando um objeto é criado.

A sua finalidade é **inicializar o objeto**, garantindo que os atributos obrigatórios sejam preenchidos corretamente.

2 O Construtor Pode Retornar Algum Valor?

 Pergunta possível:

 O construtor pode ter um tipo de retorno, como `void` ou `int`?

 **Não!** O construtor **não tem um tipo de retorno**, nem mesmo `void`.

Se colocarmos um tipo de retorno, ele **passa a ser um método normal**, não um construtor.

3 Podemos Ter Múltiplos Construtores? (Sobrecarga)

 Pergunta possível:

 O que acontece se tivermos múltiplos construtores numa classe?

 Resposta:

Sim, podemos ter **múltiplos construtores com diferentes parâmetros**. Isto chama-se **sobrecarga de construtores (constructor overloading)**.

4 O Que Acontece Se Não Criarmos Nenhum Construtor?

 Pergunta possível:

 *O que acontece se não definirmos um construtor na classe?*

 Resposta:

Se **não criarmos nenhum construtor**, o Java **gera automaticamente um construtor padrão vazio**.

 **Mas se criarmos um construtor com parâmetros, o construtor padrão deixa de existir!**

5 O Construtor Pode Ser Privado? Para Que Serve?

 Pergunta possível:


 *Podemos ter um construtor privado? Para que serve?*

 Resposta:

Sim, podemos criar um **construtor privado**. Isto é útil quando queremos **impedir que objetos sejam criados diretamente** ou, por exemplo, quando queremos garantir que só **existe um único objeto** dessa classe em toda a aplicação.

6 O Construtor Pode Chamar Outro Construtor? (**this()**)

 Pergunta possível:

 *Como podemos reaproveitar um construtor dentro de outro?*

 Resposta:

Usamos `this()` para chamar outro construtor da mesma classe.

 **Exemplo:**

```
java
CopiarEditar
class Produto {
    private String nome;
    private double preco;
```

```
public Produto(String nome) {  
    this(nome, 0.0); // Chama o outro construtor  
}  
  
public Produto(String nome, double preco) {  
    this.nome = nome;  
    this.preco = preco;  
}  
}
```

✅ Agora, criar `Produto("Celular")` chama automaticamente o segundo construtor!