
FIAP

Domain Driven Design
using Java

AGENDA

1. Construtores, Sobrecarga e Encapsulamento
2. Exercícios

Introdução

- Java, por ser uma linguagem de programação orientada a objetos, utiliza diversos recursos para estruturar e organizar código.
- Neste estudo, abordaremos Construtores, Sobrecarga de Métodos e Encapsulamento.

Construtores

- Construtores são métodos especiais que inicializam o estado de um objeto
- São chamados automaticamente quando um objeto é instanciado.

- Características:

- Nome do construtor = nome da classe.
- Não possuem tipo de retorno.
- Podem ser sobrecarregados.
- Servem para inicializar valores dos atributos ou realizar operações na criação do objeto.

Construtores - Exemplo

```java
public class Carro {
    private String marca;
    private String modelo;
    private int ano;

    // Construtor sem parâmetros
    public Carro() {
        this.marca = "Sem marca";
        this.modelo = "Sem modelo";
        this.ano = 0;
    }

    // Construtor com parâmetros
    public Carro(String marca, String modelo, int ano) {
        this.marca = marca;
        this.modelo = modelo;
        this.ano = ano;
    }
}
```

Sobrecarga de Métodos

- A sobrecarga de métodos permite que uma classe possua múltiplos métodos com o mesmo nome, mas com diferentes parâmetros.

- Características:

- A assinatura do método deve ser diferente (número, tipo ou ordem dos parâmetros).
- O compilador decide qual método chamar com base nos argumentos fornecidos.

Sobrecarga de Métodos - Exemplo

```java
public class Calculadora {
    // Método para somar dois inteiros
    public int somar(int a, int b) {
        return a + b;
    }

    // Método para somar dois decimais
    public double somar(double a, double b) {
        return a + b;
    }

    // Método sobrecarregado para somar três inteiros
    public int somar(int a, int b, int c) {
        return a + b + c;
    }
}
```

Encapsulamento

- Encapsulamento é o conceito de ocultar detalhes internos da implementação e expor apenas o necessário.

- Benefícios:

- Melhora a modularidade e reutilização de código.
- Protege os dados de acesso indevido.
- Permite mudanças internas sem afetar o restante do código.

- Como aplicar:

- Use modificadores de acesso (*private*, *public*).
- Crie métodos *getter* e *setter* para acessar e modificar os atributos privados.

Encapsulamento - Exemplo

```java
public class Pessoa {
    private String nome;
    private int idade;

    // Métodos getter e setter para 'nome'
    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    // Métodos getter e setter para 'idade'
    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }
}
```

Exercícios

1. Crie uma classe chamada Livro com os atributos:
   - titulo (String), autor (String), desconto(double) e preco (double).

2. Implemente:
   - Um construtor com todos os parâmetros;
   - Um construtor sem parâmetros que inicialize os atributos com valores padrão;
   - Métodos *getter* e *setter* para todos os atributos;
   - Um método sobrecarregado para aplicar descontos:
   - Um desconto em porcentagem (desconto(double percentual)).
   - Um desconto em valor fixo (desconto(double valorFixo)).

Exercícios

3. No programa principal:
   - Crie três objetos de Livro.
   - Deixe o atributo desconto com visibilidade apenas na classe Livro.
   - Aplique diferentes tipos de desconto em cada livro e exiba os resultados.


FIAP