Guia Java — Parte 2: Conceitos Avançados

Esta é a segunda parte do guia básico de Java. Aqui você aprenderá conceitos mais avançados e fundamentais para compreender como o Java trabalha internamente com orientação a objetos e reaproveitamento de código.

Construtor: É um método especial usado para inicializar objetos. Ele tem o mesmo nome da classe e não possui tipo de retorno. É chamado automaticamente quando um objeto é criado com 'new'. Exemplo: public class Pessoa { String nome; public Pessoa(String nome) { this.nome = nome; } }

Sobrecarga de métodos (Overloading): Acontece quando há vários métodos com o mesmo nome, mas com diferentes parâmetros. O compilador escolhe qual método usar conforme os argumentos passados. Exemplo: public void somar(int a, int b) public void somar(double a, double b)

Sobrescrita de métodos (Overriding): É quando uma subclasse redefine um método que já existe na superclasse. Serve para alterar o comportamento herdado. Exemplo: @Override public void falar() { System.out.println("O cachorro está latindo!"); }

Herança: Permite que uma classe (filha) herde atributos e métodos de outra (pai). A palavra-chave usada é 'extends'. Isso ajuda a reutilizar código e criar hierarquias. Exemplo: public class Animal {} public class Cachorro extends Animal {}

Polimorfismo: Permite que um mesmo método tenha comportamentos diferentes dependendo do tipo do objeto. Ocorre principalmente quando usamos herança e sobrescrita de métodos.

Interface: É um contrato que define métodos que uma classe deve implementar. As interfaces são criadas com a palavra-chave 'interface'. Exemplo: public interface Animal { void emitirSom(); }

Abstração: É o processo de esconder detalhes complexos e mostrar apenas o necessário. Em Java, isso é feito usando classes abstratas e interfaces.

Encapsulamento: É o princípio de proteger os dados de uma classe, tornando seus atributos privados e permitindo o acesso apenas por meio de métodos públicos (getters e setters).

Pacotes (packages): Servem para organizar o código em grupos lógicos. Cada pacote representa um diretório. São declarados no topo do arquivo com 'package nomeDoPacote;'.

Import: Usado para trazer classes de outros pacotes para serem utilizadas no arquivo atual. Exemplo: import java.util.List;

Compreender esses conceitos é essencial para dominar a Programação Orientada a Objetos em Java. Eles formam a base para desenvolvimento de sistemas robustos, APIs e frameworks modernos como Spring Boot e JavaFX.