Super Codador Java : Desperte o Dev Saiyajin

Métodos Java

Por Que Métodos São a Alma do Código Java?

Se você já escreveu mais de 10 linhas de código em Java, provavelmente percebeu que repetir comandos não é nada produtivo — e pode virar uma bagunça rapidamente. É aqui que entram os métodos: pequenos blocos de código reutilizáveis que deixam seus programas mais organizados, limpos e fáceis de manter.

Neste capítulo, você vai aprender:

- Como criar métodos simples, com e sem parâmetros;
- Como retornar valores de um método;
- A diferença entre métodos estáticos e não estáticos;
- Como usar vários tipos de parâmetros no mesmo método;
- E como aplicar a sobrecarga de métodos (overloading) para dar flexibilidade ao seu código.

Tudo será explicado com exemplos práticos e diretos, sempre pensando no que você pode aplicar de verdade no seu dia a dia como programador Java.

Vamos nessa? Seu código está prestes a virar Super Sayajin da organização.

01

Método Simples: A Base de Tudo

Método Simples: A Base de Tudo

Métodos simples são funções que não recebem parâmetros nem retornam valores. Usamos quando queremos organizar comandos que se repetem.

JAVA SAYAJIN - GABRIEL.java

```java
public class Saudacao {
    public static void dizerOla() {
        System.out.println("Olá, seja bem-vindo ao mundo Java!");
    }

    public static void main(String[] args) {
        dizerOla(); // Chamada do método
    }
}
```

👉 Uso real: Mostrar mensagens padrão em vários pontos do programa.

02

Método com Parâmetros: Personalize as Ações

Método com Parâmetros: Personalize as Ações

Parâmetros permitem passar dados para o método, tornando-o reutilizável e dinâmico.

JAVA SAYAJIN - GABRIEL.java

```java
public class Calculadora {
    public static void somar(int a, int b) {
        System.out.println("Resultado: " + (a + b));
    }

    public static void main(String[] args) {
        somar(10, 5); // Resultado: 15
    }
}
```

👉 Uso real: Operações matemáticas, configurações, personalizações de exibição.

03

Método com Retorno: Recebendo Valores de Volta

Método com Retorno: Recebendo Valores de Volta

Quando queremos que o método devolva um valor, usamos **return**.

JAVA SAYAJIN - GABRIEL.java

```java
public class Conversor {
    public static double celsiusParaFahrenheit(double c) {
        return (c * 9/5) + 32;
    }

    public static void main(String[] args) {
        double temperatura = celsiusParaFahrenheit(25);
        System.out.println("Temperatura em Fahrenheit: " + temperatura);
    }
}
```

👉 Uso real: Conversões, cálculos, verificações, etc.

04

Método com Vários Tipos de Parâmetros

Método com Vários Tipos de Parâmetros

Podemos combinar tipos diferentes de dados no mesmo método.

JAVA SAYAJIN - GABRIEL.java

```java
public class Mensagem {
    public static void exibirMensagem(String nome, int idade) {
        System.out.println("Olá " + nome + ", você tem " + idade + " anos.");
    }

    public static void main(String[] args) {
        exibirMensagem("Carlos", 30);
    }
}
```

👉 Uso real: Entrada de dados de usuário, personalização de sistemas.

05

Método Estático vs Não Estático

Método Estático vs Não Estático

static: pode ser chamado sem criar um objeto.

não static: precisa criar um objeto da classe.

JAVA SAYAJIN - GABRIEL.java

```java
public class Util {
    public static void metodoEstatico() {
        System.out.println("Chamado sem precisar de objeto.");
    }

    public void metodoNormal() {
        System.out.println("Chamado por meio de um objeto.");
    }

    public static void main(String[] args) {
        metodoEstatico(); // OK

        Util u = new Util();
        u.metodoNormal(); // OK
    }
}
```

👉 Uso real: Utilitários e ferramentas (estático), comportamentos personalizados de objetos (não estático).

06

Sobrecarga de Métodos (Overloading)

Sobrecarga de Métodos (Overloading)

Mesma função com diferentes parâmetros.

JAVA SAYAJIN - GABRIEL.java

```java
public class Saudacao {
    public static void ola() {
        System.out.println("Olá!");
    }

    public static void ola(String nome) {
        System.out.println("Olá, " + nome + "!");
    }

    public static void main(String[] args) {
        ola(); // Olá!
        ola("João"); // Olá, João!
    }
}
```

👉 Uso real: Flexibilizar funções para diferentes necessidades.

Agradecimentos!

Obrigado por ler até aqui!

Muito obrigado por ter lido este eBook!

Esperamos que o conteúdo tenha te ajudado a entender melhor os fundamentos da construção de métodos em Java, de forma prática e objetiva.

Este material foi gerado por Inteligência Artificial com foco exclusivo em estudo e treinamento, sem fins comerciais. A ideia é oferecer uma base rápida, acessível e clara para quem está dando os primeiros passos — ou reforçando conhecimentos — no universo da programação com Java.

Continue praticando, explorando e evoluindo. O aprendizado em tecnologia é constante, e cada linha de código te aproxima ainda mais da sua próxima conquista.

Bons estudos e até a próxima!