Métodos

FUNDAMENTOS DE PROGRAMAÇÃO

Aulas finais de java! Presente de Despedida.☺

APRESENTAÇÃO DO CONTEÚDO

O que iremos ver:

- O que são métodos
- Métodos com retorno e sem retorno
- Parâmetros
- Modificador static
- Escopo de variáveis
- Exercícios propostos

INTRODUÇÃO

O que são Métodos?

Métodos são blocos de código que executam uma tarefa específica quando são chamados. Eles funcionam como "mini-programas" dentro do seu programa principal, ajudando a organizar o código e evitar repetição.

Vantagens de usar métodos:

Organização: Código mais limpo e fácil de entender

Reutilização: Escreva uma vez, use várias vezes

Manutenção: Mais fácil corrigir e atualizar o código

SINTAXE E EXEMPLO

Sintaxe:

```
<modificadorAcesso> <modificadorStatic> <tipoRetorno> <nomeMetodo>(<parametros>) {
    // corpo do método
    // código a ser executado
}
```

Exemplo:

```
public static void saudar() {
    System.out.println("Olá, bem-vindo!");
}
```

MÉTODOS COM RETORNO E SEM RETORNO

Métodos sem retorno (void)

Usamos **void** quando o método executa uma acção mas não precisa devolver nenhum valor.

```
public class Exemplo {
    // Método que apenas imprime, não retorna nada
    public static void imprimirMensagem(String nome) {
        System.out.println("Olá, " + nome + "!");
    }
    public static void main(String[] args) {
        imprimirMensagem("João");  // Saída: Olá, João!
    }
}
```

MÉTODOS COM RETORNO E SEM RETORNO

Métodos com retorno

Quando o método precisa **devolver um valor**, especificamos o tipo (int, double, String, etc.).

```
 // Método que retorna um número inteiro
    public static int somar(int a, int b) {
        int resultado = a + b;
        return resultado;  // OBRIGATÓRIO usar return
    }
```

```
// Método que retorna texto
public static String obterSaudacao(String nome) {
    return "Bem-vindo, " + nome;
}

public static void main(String[] args) {
    int soma = somar(5, 3);
    System.out.println("Soma: " + soma);

    String mensagem = obterSaudacao("Maria");
    System.out.println(mensagem);
}
```

PARÂMETROS

Parâmetros são informações passadas aos sub-programas (são valores que, na linha de chamada, ficam entre os parênteses e que estão separados por vírgulas).

A quantidade de parâmetros após a sua declaração não pode ser alterada, bem como sua sequência e respectivos tipos.

Sintaxe:

```
<tipoRetorno> <nomeMetodo>(<tipo1> <parametro1>, <tipo2> <parametro2>, ...) {
    // usar parametro1 e parametro2 aqui
}
```

PARÂMETROS(EXEMPLO)

```
// Método sem parâmetros
public static void dizerOla() {
  System.out.println("Olá!");
}
// Método com 1 parâmetro(nome)
public static void dizerOlaPersonalizado(String nome) {
  System.out.println("Olá, " + nome + "!");
}
// Método com múltiplos parâmetros(nota1, nota2 e nota3)
public static int calcularMedia(int nota1, int nota2,
int nota3) {
 int soma = nota1 + nota2 + nota3;
 return soma / 3;
}
```

```
public static void main(String[] args) {
   dizerOla(); // Olá!
   dizerOlaPersonalizado("Ana"); // Olá, Ana!
   int media = calcularMedia(8, 7, 9);
   System.out.println("Média: " + media); //
Média: 8

}
```

MODIFICADOR STATIC

O MODIFICADOR STATIC

O modificador **static** indica que um método pertence à **classe**, não a um objeto específico.

Entendendo o main

```
public static void main(String[] args) {
    // código aqui
}
```

Vamos entender cada parte:

public: Precisa ser público para a JVM (Java Virtual Machine) poder chamar

static: A JVM precisa chamar o método sem criar um objeto da classe

void: O método não retorna nenhum valor

main: Nome especial que a JVM procura para iniciar o programa

String[] args: Array de argumentos que podem ser passados ao programa

ESCOPO DE VARIÁVEIS

Variável Global:

- São as variáveis declaradas no início da classe, é, podemos declarar variáveis no início da classe e são chamadas de atributos em um programa.
- Pode ser usada por qualquer sub-programa(método) subordinada ao programa principal.

Variável Local:

- Declarada dentro de um subprograma e só é válida dentro da própria rotina/sub-programa.
- Após o final de cada execução do sub-programa, as variáveis locais são destruídas.

VARIÁVEIS GLOBAIS(EXEMPLO)

```
public class Contador {
    // Variável global (de classe)
    private static int total = 0;
    public static void incrementar() {
        total++;  // Pode acessar total
        System.out.println("Total: " + total);
    }
    public static void main(String[] args) {
        incrementar();  // Total: 1
        incrementar();  // Total: 2
        incrementar();  // Total: 3
    }
}
```

VARIÁVEIS LOCAIS(EXEMPLO)

```
public class ExemploEscopo {

    public static void metodo1() {
        int x = 10;  // variável local de metodo1
        System.out.println(x);  // Funciona
    }

    public static void metodo2() {
        // System.out.println(x);  // ERRO! x não existe aqui
        int y = 20;  // variável local de metodo2
    }

    public static void main(String[] args) {
        // System.out.println(x);  // ERRO! x não existe aqui
        // System.out.println(y);  // ERRO! y não existe aqui

        int z = 30;  // variável local de main
        System.out.println(z);  // Funciona
    }
}
```

FIM

Feito por:

- Anselmo Nhamage