#Reskilling4Employement
Software Developer

Programação Estruturada

Métodos - Funções - Procedimentos

Vitor Santos

Conteúdo

- Métodos
- Procedimentos
- Variáveis Locais vs Globais

Métodos

Funções = método

- **Funções** são sequências de instruções que aceitam **argumentos** (definidos na lista de parâmetros da função) e que devolvem um **resultado.**

- Têm o propósito de tornar a escrita de programas mais estruturada e mais fácil de ler.

- Além de permitirem uma grande reutilização de porções de códigos, aumentam a legibilidade do programa.

Métodos - Exemplos da Livraria de Java

- **main( )** é uma função que é usada em todos os programas e é executada logo que o programa arranca. Recebe um parâmetro (array do tipo String) e não devolve nada (tipo void).
- **sqrt(25)** é uma função que calcula a raiz quadrada de um número. Recebe um parâmetro (um valor numérico) e devolve um valor numérico. Faz parte da livraria Math.
- **random( )** é uma função que gera um número real aleatório entre 0.0 e 1.0, não recebe parâmetros.

Sintaxe do Método em Java

public static boolean numPrimo (int num) { EXEMPLO

```
public static tipo_da_função nome_da_funcao (lista_de_parâmetros) {
    declaração_de_variáveis_locais;
    bloco_instruções;
    return (valor_ou_expressão);
}
```

Sintaxe do Método em Java

- Tipos da função
  - int, double, void, boolean, etc...
- Nome da Função
  - Seguir as mesmas regras de nomes de variáveis
- Lista de Argumentos
  - São os parâmetros que a função recebe de "quem" a invocou. Os parâmetros vão funcionar como variáveis locais da função. É preciso indicar o nome e tipo de cada parâmetro. Quando se invoca uma função, o número, tipo e ordem dos parâmetros enviados deve ser coincidente com os parâmetros na declaração da função

Sintaxe do Método em Java

- **Declaração de variáveis locais**
  - Declaração das variáveis que serão usadas no corpo da função
- **Resultado:** Termina a função e devolve o seu resultado a "quem" a invocou. O resultado pode ser um valor (exemplo: um número) ou uma expressão (exemplo: uma variável ou um cálculo matemático). Representa-se escrevendo **return** seguido do valor ou expressão a devolver. Uma função pode ter várias intruções de devolução de resultados, no entanto, apenas uma delas é executada em cada invocação da função.

Exemplo de um Método em Java

```
public static int modulo_diferenca_inteira (int a, int b) {
    int resultado;

        10-2
    if (a > b){
        resultado = a - b;   =8
    }
    else{
        resultado = b - a;
    }
    return resultado;
}
```

peço 2 parâmetros int a e b
que são usados como variáveis e
vão ter um valor.

int a=10 int b=2

Exemplo da invocação de um Método em Java

```
public static void main(String[] args) {

    int num1, num2, res;

    System.out.print("Introduza um número: ");
    num1 = input.nextInt( );

    System.out.print("Introduza um número: ");
    num2 = input.nextInt( );

    res = modulo_diferenca_inteira (num1, num2);

    System.out.print("Módulo Dif. Int.: " + res);

}
```

Procedimento

- Função com tipo de dados de saída "void" (que significa não devolver qualquer tipo de dados) chamamos procedimento. Pode incluir, contudo, uma lista de parâmetros.
- Exemplo:

```
public static void copyright ( ) {
    System.out.println("*******************************");
    System.out.println("* Programa Realizado por: *");
    System.out.println("* Formador - Vitor Santos *");
    System.out.println("*******************************");
}
```

Variáveis Locais vs Globais

- As **variáveis locais** são declaradas dentro do corpo de uma função. Para o programa essas variáveis são apenas "visíveis" dentro da própria função e "destroem-se" quando a função termina.

- Desde que estejam declaradas em funções distintas, podem existir variáveis locais com o mesmo nome, não existindo qualquer relação entre elas.

- Depois de terminada uma função, são eliminadas todas as suas variáveis locais.

Variáveis Locais vs Globais

- As **variáveis globais** são declaradas logo no início do programa antes de qualquer função, sendo "visíveis" em todo o programa. Qualquer alteração aos seus valores repercute-se em todo o programa.
- Uma **variável local** com o mesmo nome de uma variável global **tem prioridade** sobre esta última durante a execução da função onde a variável local foi declarada.
- **Sempre que possível devem usar-se as variáveis locais**, evitando assim eventuais efeitos colaterais que ocorrem quando se usam variáveis globais.
- À "região de influência" de uma variável chamamos âmbito (do inglês, ***scope***).

Resumo e Boas Práticas

- As **funções evitam a repetição de código** ao longo do programa e reduzem a sua complexidade.
- Quando uma função é invocada, a função que a invoca é "suspensa" temporariamente. De seguida são executadas as instruções da função invocada. Quando esta termina, o controlo da execução do programa volta ao ponto onde foi invocada a função.
- Ao invocar uma função são enviados argumentos que serão recebidos nos parâmetros da função. Estes serão armazenados em variáveis locais que são automaticamente inicializadas com os valores enviados.

Resumo e Boas Práticas

- Depois de terminar o seu funcionamento, uma função pode devolver um valor para a função que a invocou.
- Cada função, tal como uma variável, tem que ter um **nome único**, que permite a mesma ser invocada em qualquer ponto do programa a que pertence.
- Uma função, em geral, deve efetuar **apenas uma tarefa bem definida**.

Resumo e Boas Práticas

- Uma função, em geral, deve sempre retornar o valor ao invés de só apresentar na consola.
- O código de uma função deve ser o mais **independente** possível do restante do código do programa. Se possível, deve também ser o mais genérico possível para que esta possa ser reutilizada noutros programas.
- Como vimos, uma função que devolve "void" chamamos de procedimento.

#Reskilling4Employement
Software Developer

Programação Estruturada

Métodos - Funções - Procedimentos

Vitor Santos