Funções(métodos) em Java.

****

Definindo funções

Em Java, uma função é composta por **tipo de retorno**, **nome**, **parâmetros** e **implementação**:

- **Tipo de Retorno**: Indica qual é o tipo de dado que é entregue como resposta pela função. Pode ser qualquer tipo de dado utilizado na declaração de uma variável, salvo o tipo void, que é responsável por indicar que a função apenas executa o seu código mas não entrega uma resposta.
- **Nome**: Utilizado para referenciar a função posteriormente.
- **Parâmetros**: Indicam quais informações devem ser entregues à função para que ela execute. Por exemplo, a função square, que eleva um número ao quadrado, necessita saber *qual* número será elevado ao quadrado.
- **Implementação**: Define a sequência de código que será executada ao se chamar a função. "Chamar" uma função significa pedir para que sua implementação seja executada.

exemplos

```
int square(int x) {
   return x * x;
}
```

Descrevendo os elementos dessa função:

- **Tipo de Retorno**: inteiro
- **Nome**: square.
- **Parâmetros**: int x. Ou seja, essa função exige que seja enviado um valor do tipo int como primeiro parâmetro, e internamente ele será chamado de x.
- **Implementação**: A implementação apenas significa "retorne a execução para quem chamou a função e entregue x * x como resposta".

****

Existem algumas regras básicas para criar funções (também chamada de métodos em Java).

As boas práticas de programação dizem que o ideal para criar funções em java é definir um bom nome, simples ou composto, contendo verbos. Por exemplo:

public static void somar(int x, int y) {}

Neste exemplo, temos que o nome da função é somar, ou seja, um verbo. Outra coisa que merece atenção é a responsabilidade da função. **Toda função tem que ser específica.**

Ela deve resolver um único problema, deve ser específica. Isso é o que faz um programa ser modular, ser fácil de realizar manutenção, ser fácil de atualizar, pois você tem que resolver uma responsabilidade específica, e isso ajuda a programar.

Caso você precise nomear uma função com nomes compostos, opte por nomes simples e bem descritivos, tal como:

```
public static void converterNomesParaMaiusculo(String nome) {}
```

Assim, fica fácil de você revisitar o código entender, e outros programadores também entendem o seu código, para ser legível, ser entendível.

Outro ponto que pode ser importante é *o tipo de retorno da função*, se ela é com ou sem retorno. Quando você ver a palavra *"void" é porque a função é sem retorno.*

Quando a função tem retorno, ela é representado pelo tipo do dado que você quer retornar, que normalmente, é um tipo primitivo: int, float, double, boolean ou ainda, pode ser um objeto. Neste caso, o tipo de retorno é o nome da classe.

```
public static int somar(int x, int y) {

        return  x + y;

}
```

Assim, você tem a visão completa, sabendo que a função com retorno tem duas coisas como características principais, que são: O tipo do retorno, que falei acima, e a palavra reservada return.

Não deixe a função ficar muito grande. Lembre-se de deixá-la com responsabilidades únicas. Afinal, você pode conectar várias funções e assim, criar um sistema modular, limpo, objetivo, que no final, transforma-se em um sistema.

Exemplo prático

```
public class Funcoes {

    public static void saudacao() {
        System.out.println("Olá, mundo!");
    }

    public static int soma(int a, int b) {
        return a + b;
    }
    public static void main(String[] args) {
                    saudacao();
        int resultado = soma(5, 3);
        System.out.println("Resultado da Soma: " + resultado);
    }

}
```

A linguagem Java está equipada com uma biblioteca abrangente para funções e operações matemáticas. Ela é chamada de classe "Math" e reside no pacote java.lang . A biblioteca Math inclui métodos para operações numéricas fundamentais, trigonometria, encontrar min-max, gerar números aleatórios e operações logarítmicas.

Funções matemáticas em Java: https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

Aritmética: https://pet-comp-ufsc.github.io/tutorials/langs/java/others/arithmetic-details.html

Exemplo: Elevando um número a uma potência em Java usando Math.pow

Vamos encontrar o valor de 5^4 usando Math.pow .

```
import java.lang.Math;
public class MyClass{
   public static void main(String []args){
     double answer = Math.pow(5, 4);
     System.out.println("5 raised to the power of 4 = " + answer);
   }
}
```

A saída é 625.0

Podemos converter o número para um número inteiro da seguinte maneira:

```
int answer = (int) Math.pow(5, 4);
```

Agora o resultado é 625.

Vamos elevar um número a uma potência negativa e comparar os resultados. Para este exemplo, escolheremos 4 como base e -2 como expoente.

```
import java.lang.Math;

public class MyClass{

    public static void main(String []args){
      double answer = Math.pow(4, -2);
      System.out.println(answer);
    }
}
```

Obtemos a saída 0,0625

```
public class constantes {

    public static void main(String[] args){

    System.out.println("O valor de pi é: " + Math.PI);

    System.out.println("O valor de E é: " + Math.E);

  }
```

}

Exponencial e potenciação na classe Math

Para calcular valores do tipo: e^x
usamos o método: exp() , que recebe um double e retorna um double.

numero = Math.exp(argumento)

Para calcular qualquer tipo de potências, da forma: a^b
onde a e b são do tipo double, usamos o método pow() da classe Math

numero = Math.pow(a,b)

```
public class Mathtest {


  public static void main(String[] args){

    System.out.println("'e' elevado ao quadrado = "+ Math.exp(2));

    System.out.println("2 elevado ao cubo = " + Math.pow(2, 3));

  }

}
```

Lembre-se que estes métodos retornam double.
Caso você tenha declarado um float e queira receber o resultado no sua variável float, use o cast.
Por exemplo, no caso dos nossos methods para o cálculo de IMC, o 'quadrado' retorna um float, então fazemos:
(float)Math.pow(altura,2);

Assim, o método Math.pow() irá retorna um float, ao invés do double.

Funções Arredondamento

Arredondar números é uma operação comum em muitos domínios, especialmente quando trabalhamos com finanças ou precisão específica:

- **Math.floor(value)**: Retorna o maior número inteiro menor ou igual ao valor.
- **Math.ceil(value)**: Retorna o menor número inteiro maior ou igual ao valor.

Outros Métodos Importantes

- **Math.abs(value)**: Retorna o valor absoluto.
- **Math.sqrt(value)**: Encontra a raiz quadrada.
- **Math.pow(a, b)**: Eleva "a" à potência de "b".

Algumas funções matemáticas em Java

Math.abs() - Ele retornará o valor absoluto do valor fornecido.

Math.max() - Ele retorna o Maior de dois valores.

Math.min() - Ele é usado para retornar o menor de dois valores.

Math.round() - É usado para arredondar os números decimais para o valor mais próximo.

Math.sqrt() - É usado para retornar a raiz quadrada de um número.

Math.cbrt() - É usado para retornar a raiz cúbica de um número.

Math.pow() - Ele retorna o valor do primeiro argumento elevado à potência do segundo argumento.

Math.sign() - É usado para encontrar o sinal de um determinado valor.

Math.ceil() - É usado para encontrar o menor valor inteiro maior ou igual ao argumento ou número inteiro matemático.

Math.copySign() - É usado para encontrar o valor absoluto do primeiro argumento junto com o sinal especificado no segundo argumento.

Math.nextAfter() - É usado para retornar o número de ponto flutuante adjacente ao primeiro argumento na direção do segundo argumento.

Math.nextUp() - Ele retorna o valor de ponto flutuante adjacente a d na direção do infinito positivo.

Math.nextDown() - Ele retorna o valor de ponto flutuante adjacente a d na direção do infinito negativo.

Math.floor() - É usado para encontrar o maior valor inteiro que é menor ou igual ao argumento e é igual ao número inteiro matemático de um valor duplo.

Math.floorDiv() - É usado para encontrar o maior valor inteiro que é menor ou igual ao quociente algébrico.

Math.random() - Retorna um valor duplo com sinal positivo, maior ou igual a 0,0 e menor que 1,0.

Math.rint() - Ele retorna o valor duplo mais próximo do argumento fornecido e igual ao número inteiro matemático.

Math.hypot() - Ele retorna sqrt($x^2 + y^2$) sem estouro intermediário ou estouro insuficiente.

Math.ulp() - Ele retorna o tamanho de uma unidade em último lugar do argumento.

Math.getExponent() - É usado para retornar o expoente imparcial usado na representação de um valor.

Math.IEEEremainder() - É usado para calcular a operação restante em dois argumentos conforme prescrito pelo padrão IEEE 754 e retorna o valor.

Math.addExact() - Ele é usado para retornar a soma de seus argumentos, lançando uma exceção se o resultado ultrapassar um int ou long.

Math.subtractExact() - Ele retorna a diferença dos argumentos, lançando uma exceção se o resultado estourar um int.

Math.multiplyExact() - É usado para retornar o produto dos argumentos, lançando uma exceção se o resultado ultrapassar um int ou long.

Math.incrementExact() - Ele retorna o argumento incrementado em um, lançando uma exceção se o resultado estourar um int.

Math.decrementExact() - Ele é usado para retornar o argumento decrementado em um, lançando uma exceção se o resultado estourar um int ou long.

Math.negateExact() - É usado para retornar a negação do argumento, lançando uma exceção se o resultado ultrapassar um int ou long.

Math.toIntExact() - Ele retorna o valor do argumento longo, lançando uma exceção se o valor ultrapassar um int.

Métodos Logarítmicos

Math.log() - Ele retorna o logaritmo natural de um valor duplo.

Math.log10() - É usado para retornar o logaritmo de base 10 de um valor duplo.

Math.log1p() - Ele retorna o logaritmo natural da soma do argumento e 1.

Math.exp() - Ele retorna E elevado à potência de um valor duplo, onde E é o número de Euler e é aproximadamente igual a 2,71828.

Math.expm1() - É usado para calcular a potência de E e subtrair um dela.

Métodos Trigonométricos

Math.sin() - É usado para retornar o valor do Seno trigonométrico de um valor duplo dado.

Math.cos() - É usado para retornar o valor do cosseno trigonométrico de um valor duplo dado.

Math.tan() - É usado para retornar o valor da tangente trigonométrica de um valor duplo dado.

Math.asin() - É usado para retornar o valor trigonométrico do seno do arco de um dado valor duplo

Math.acos() - É usado para retornar o valor trigonométrico do cosseno do arco de um valor duplo dado.

Math.atan() - É usado para retornar o valor trigonométrico da tangente do arco de um valor double dado.

Métodos Hiperbólicos

Math.sinh() - É usado para retornar o valor trigonométrico do cosseno hiperbólico de um valor duplo dado.

Math.cosh() - É usado para retornar o valor trigonométrico do Seno Hiperbólico de um valor duplo dado.

Math.tanh() - É usado para retornar o valor trigonométrico da tangente hiperbólica de um valor double dado.

Métodos Angulares

Math.toDegrees() - É usado para converter o ângulo especificado em Radianos para o ângulo equivalente medido em Graus.

Math.toRadians() - É usado para converter o ângulo especificado em Graus para o ângulo equivalente medido em Radianos.

****

Assinatura de uma função

Uma função é identificada pela sua **assinatura**. A assinatura de uma função é composta pelo **nome e tipos dos parâmetros** (observe que **o tipo de retorno *NÃO* faz parte da assinatura**). Assim, o compilador é capaz de diferenciar duas funções apenas analisando a assinatura delas. Por exemplo:

```
void foo() { ... }

void foo(int x) { ... } // OK, outra função

void foo(int otherParamName) { ... } // Erro: foo(int) já foi definida

int foo() { ... } // Erro: foo() já foi definida

int foo(int x) { ... } // Erro: foo(int) já foi definida
```

****

Mais exemplos.

Funções com Múltiplos Retornos: A função divide retorna um array contendo tanto o quociente quanto o resto da divisão. Na chamada da função, os valores são armazenados em um array e exibidos separadamente.

```
public class Divisao {
   public static int[] divide(int dividendo, int divisor) {
      int quociente = dividendo / divisor;
      int resto = dividendo % divisor;
      return new int[] { quociente, resto };
   }
public static void main(String[] args) {
      int[] resultado = divide(10, 3);
      System.out.println("Quociente: " + resultado[0] + " Resto: " + resultado[1]);
```

```
    }
}
```

Funções Variádicas: A função soma aceita um número variável de argumentos utilizando a sintaxe de parâmetro variádico. Os números são somados utilizando um loop for-each.

```
public class SomaVariadica {
    public static int soma(int... nums) {
        int soma = 0;
        for (int num : nums) {
            soma += num;
        }
        return soma;
    }
public static void main(String[] args) {
        int resultado = soma(1, 2, 3, 4, 5);
        System.out.println("Soma Variádica: " + resultado);
    }
}
```

****

Conceitos Básicos

- **Função (Método):** Um bloco de código que executa uma tarefa específica.
- **Parâmetros:** São valores que você pode passar para um método.
- **Tipo de Retorno:** Indica o tipo de valor que o método vai retornar.
- **Corpo do Método:** Onde as instruções são definidas.

Benefícios das Funções

1. **Reutilização de Código:** Evita a repetição de código.
2. **Modularidade:** Facilita a leitura e manutenção do código.
3. **Organização:** Separa o código em blocos lógicos.

Tipos de Funções

1. **Funções Sem Retorno (void):** Não retornam valor.
2. **Funções com Retorno:** Devolvem um valor.
3. **Funções com Parâmetros:** Recebem valores para processar.
4. **Funções Recursivas:** Chamam a si mesmas.

Boas Práticas

- **Nomeação:** Use nomes descritivos e siga as convenções de nomenclatura.

- **Tamanho:** Mantenha suas funções pequenas e focadas.
- **Evite Efeitos Colaterais:** Uma função deve fazer uma coisa e fazer bem feito.
- **Documentação:** Comente seu código e documente suas funções.

Conclusão

As funções (ou métodos) são fundamentais na programação Java, permitindo modularidade, reutilização e organização no código. Dominar este conceito é essencial para qualquer desenvolvedor Java.

Explicação da classe main.

fonte: https://www.devmedia.com.br/trabalhando-com-metodos-em-java/25917