



High performance. Delivered.



Application Delivery Fundamentals: Java

Module 3: Language Fundamentals – Parte 2

[illegible]

-

-

[illegible]

1

Types of Flow Control

- *Instruções sequenciais* são executadas na ordem em que são escritas;
- Esse é o *fluxo* padrão de um programa, conforme as instruções são executadas na ordem em que aparecem;

-
- ```
graph TD; Start(()) --> Decision{ }; Decision --> Process1([]); Decision --> Process2([]); Process1 --> End((())); Process2 --> End;
```



[illegible]

[illegible]

214869676820706572666f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820

- ```
Syntax:  if ( condition ) {  
           // statement(s) to be executed If condition is met  
        } else { // ( else statement is optional )  
           // statement(s) to be executed If condition is NOT met  
        }  
  
Example: int x = 1;  
        if ( x > 0 ) {  
            System.out.println( "The value of x is greater than zero" );  
        } else {  
            System.out.println( "The value of x is less than or equal to zero" );  
        }
```

Operador Ternário



-
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a pen and looking at the board. Two men in business suits are seated at a glass table in front of her, looking at the whiteboard. The man on the left is holding a laptop. The man on the right is resting his chin on his hand. The setting is a modern office with large windows in the background.

[illegible]

[illegible]

-
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a marker in her right hand. Two men in business suits are seated at a glass table in the foreground, looking towards the whiteboard. The man on the left is holding a folder, and the man on the right has his hand to his chin in a thoughtful pose. The office has large windows in the background, letting in bright light.

Syntax: `switch (exp) {`

Example: `int x = 1;`

-
- ```
graph TD; Start(()) --> D1{value1?}; D1 -- T --> B1[block1]; B1 -- break --> End((())) ; D1 -- F --> D2{value2?}; D2 -- T --> B2[block2]; B2 -- break --> End ; D2 -- F --> D3{value3?}; D3 -- T --> B3[block3]; B3 -- break --> End ; D3 -. F .-> DB[defaultBlock]; DB --> End ;
```

# Switch-Case

```
switch (expressão)
{
 case valor1:
 expressão1;
 break;
 case valor2:
 expressão2;
 break;
 default:
 expressão;
}
```

Utilizado para cobrir múltiplas escolhas sobre valores alternativos de variáveis *int*, *byte*, *short*, *long* ou *char*.



[illegible]

- ```

public class SwitchStatement3
{
    public static void main(String args[])
    {
        int num = 3;
        switch(num)
        {
            case 1:
                System.out.println("Inside Case 1");
                break;
            case 2:
                System.out.println("Inside Case 2");
                break;
            case 3:
                System.out.println("Inside Case 3");
                break;
            case 4:
                System.out.println("Inside Case 4");
                break;
            case 5:
                System.out.println("Inside Case 5");
                break;
            default:
                System.out.println("Inside default case");
        }
        System.out.println("After Switch Statement");
    }
}

```

[illegible]

-

Types of Flow Control

- Estruturas de iteração executam instruções repetidamente com base em uma condição
- Esse tipo de controle de fluxo pode ser implementado usando as diferentes instruções de loop:
 - for-loop
 - while-loop
 - do-while loop

[illegible]

[illegible]

- Syntaxe

The diagram illustrates the components of a C++ for loop. It shows the following code snippet:

```
for (int j=0; j<10; j++)  
{  
  
}
```

Three labels with arrows point to the corresponding parts of the loop header:

- Initialization expression** points to `int j=0;`
- Test Condition** points to `j<10;`
- Increment expression** points to `j++`

[illegible]

[illegible]

-
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a marker in her right hand. Two men in business suits are seated at a glass table in the foreground, looking towards the whiteboard. The man on the left is holding a folder, and the man on the right has his hand to his chin in a thoughtful pose. The office has large windows in the background, letting in bright light.

[illegible]

-
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a marker in her right hand. Two men in business suits are seated at a glass table in the foreground, looking towards the whiteboard. The man on the left is holding a folder, and the man on the right has his hand to his chin in a thoughtful pose. The office has large windows in the background, letting in bright light.

Copyright © 2023 Accenture All Rights Reserved.

- ```
Syntax: for (E element : collectionOfElement) {
 //statements here
 }

Example: for (String str : listString) {
 System.out.println (“The value of str is: “ + str);
 }
```

[illegible]

- 
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a marker in her right hand. Two men in business suits are seated at a glass table in the foreground, looking towards the whiteboard. The man on the left is holding a folder, and the man on the right has his hand to his chin in a thoughtful pose. The office has large windows in the background, letting in bright light.



- 
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a marker in her right hand. Two men in business suits are seated at a glass table in the foreground, looking towards the whiteboard. The man on the left is holding a folder, and the man on the right is resting his chin on his hand. The office has large windows in the background, letting in bright light.



```
for (int i = 0; i < collection.size(); i++) {
 System.out.println(collection.get(i));
}
```

# While Loop

- while() executa instruções repetidamente enquanto uma condição permanece verdadeira

**Syntax:**

```
while (condition) {
 //statements here
}
```

**Example:**

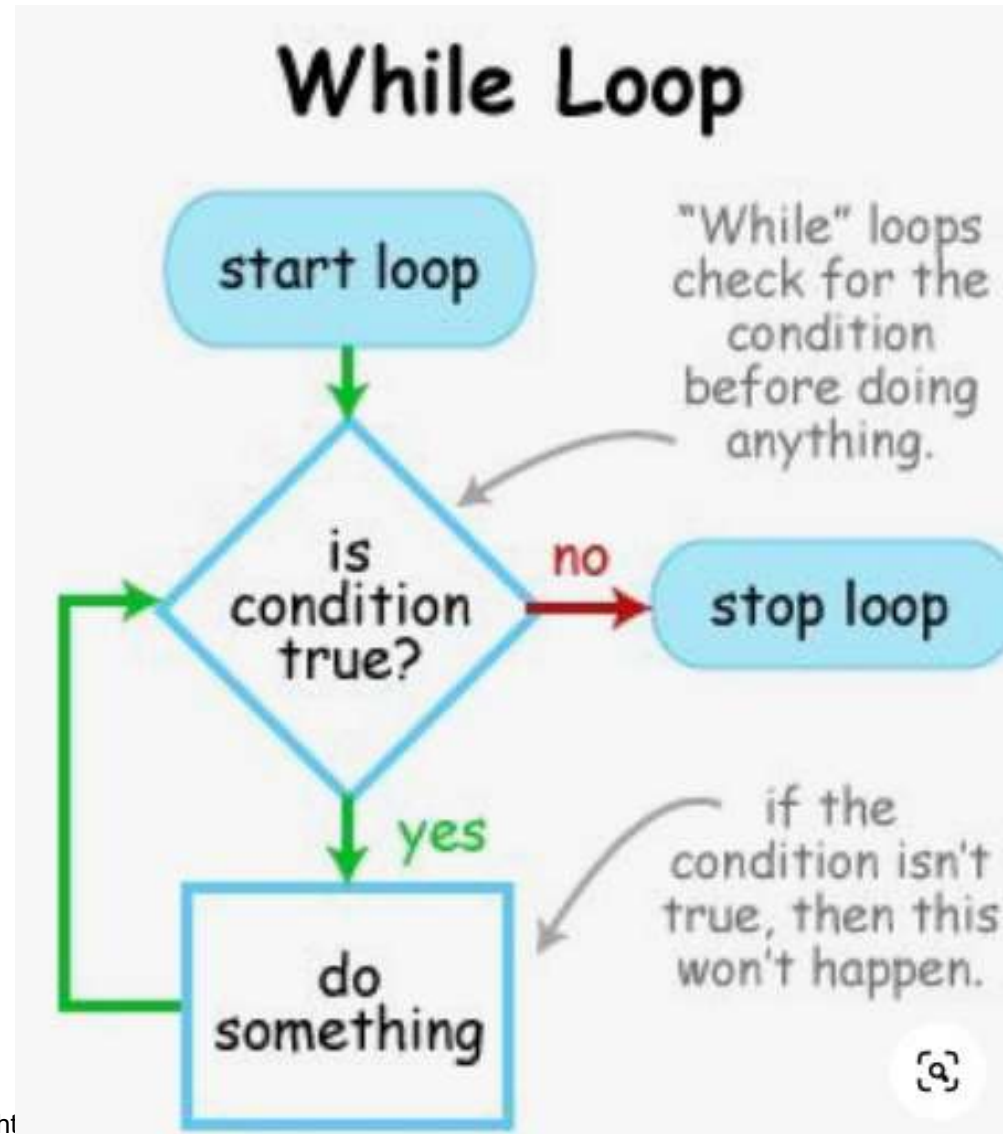
```
int x = 1;
while (x < 11) {
 System.out.println ("The value of x is: " + x);
}
```



Refer to the WhileLoopSample.java sample code.

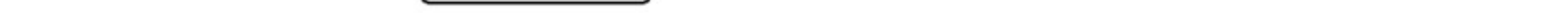
# While Loop

- while() executa instruções repetidamente enquanto uma condição permanece verdadeira



[illegible]

- 
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a marker in her right hand. Two men in business suits are seated at a glass table in the foreground, looking towards the whiteboard. The man on the left is holding a folder, and the man on the right is resting his chin on his hand. The office has large windows in the background, letting in bright light.



# While vs do .. while





[illegible]



[illegible]

Copyright © 2023 Accenture All Rights Reserved. 40

Refer to the `ScannerExample.java` sample code.

# Tipos de input

| Method                     | Description                                      |
|----------------------------|--------------------------------------------------|
| <code>nextBoolean()</code> | Reads a <code>boolean</code> value from the user |
| <code>nextByte()</code>    | Reads a <code>byte</code> value from the user    |
| <code>nextDouble()</code>  | Reads a <code>double</code> value from the user  |
| <code>nextFloat()</code>   | Reads a <code>float</code> value from the user   |
| <code>nextInt()</code>     | Reads a <code>int</code> value from the user     |
| <code>nextLine()</code>    | Reads a <code>String</code> value from the user  |
| <code>nextLong()</code>    | Reads a <code>long</code> value from the user    |
| <code>nextShort()</code>   | Reads a <code>short</code> value from the user   |

# Atividade 1

- 
- A 3D architectural rendering of the stadium's interior. The view is from an elevated angle, looking down at the green football pitch. The pitch is marked with white lines for the center circle, penalty areas, and goal lines. The seating is arranged in tiers, with the lower tiers being red and the upper tiers being blue. A large, curved white structure, possibly a scoreboard or a decorative element, is visible on the left side of the stands. The overall design is modern and functional.



## O programa LocalizaNúmero.JAVA:

1. Corrigir o bug do programa.
2. Usar o scanner.

Copyright © 2023 Accer

[illegible]

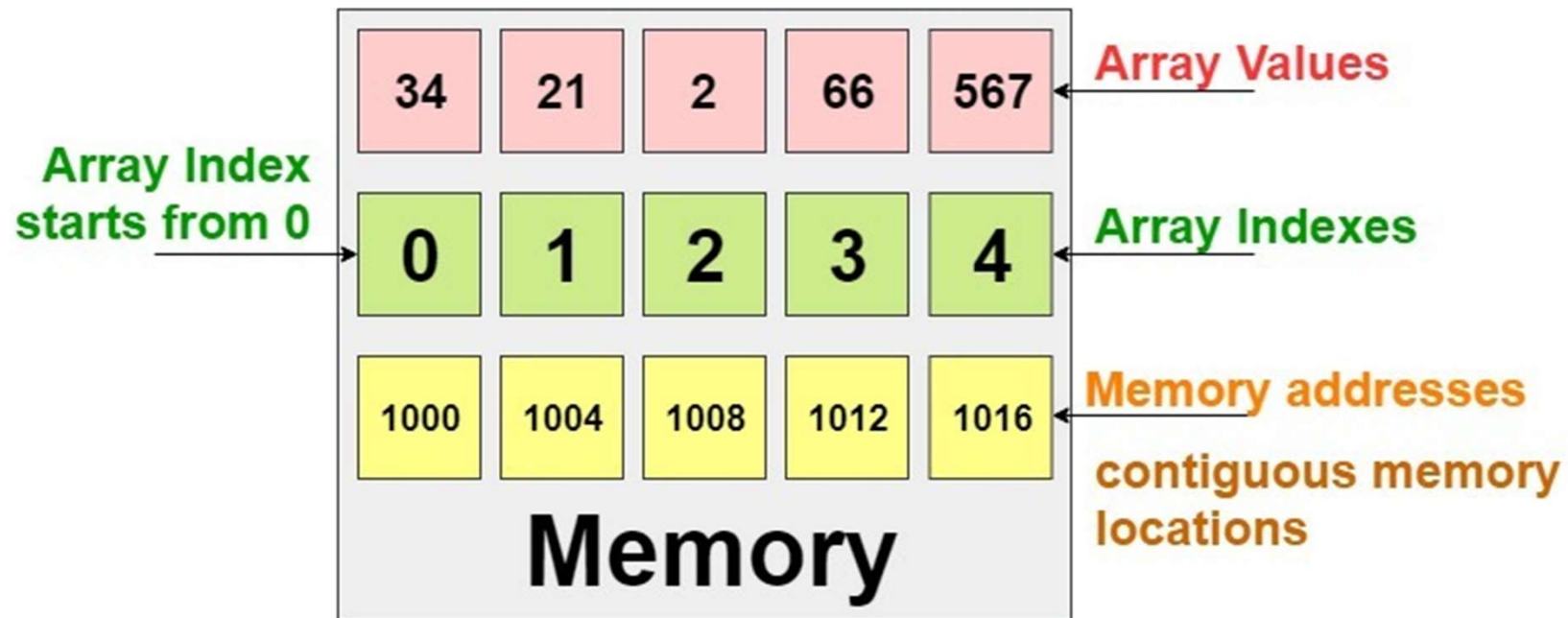
Copyright © 2023 Accenture All Rights Reserved. 45



[illegible]

- Sintaxe Básica

```
int x[] = new int[] {34, 21, 2, 66, 567};
```





[illegible]

[illegible]

[illegible]

[illegible]


- ## Como criamos um array vazio?



42 51 63 90 87

[illegible]

- NEW



- 
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a pen in her right hand. Two men in business suits are seated at a glass table in the foreground, looking at the whiteboard. The man on the left is holding a laptop. The man on the right has his hand on his chin, appearing to be in deep thought. The whiteboard has some faint diagrams and text on it. The setting is a modern office with large windows in the background.

# Arrays: Multi-Dimensional Arrays

- Uma matriz multidimensional pode ser simplesmente considerada como matrizes de matrizes;
- Cada par de colchetes representa uma dimensão da matriz;

```
int[][] arr = {
 {1, 2, 3},
 {4, 5, 6}
};
```

col  $\longrightarrow$

|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 |

row  
 $\downarrow$



Refer to the `MultiDimensionalArraySample.java` sample code.

|       | Column 1 | Column 2 | Column 3 | Column 4 |
|-------|----------|----------|----------|----------|
| Row 1 | a[0][0]  | a[0][1]  | a[0][2]  | a[0][3]  |
| Row 2 | a[1][0]  | a[1][1]  | a[1][2]  | a[1][3]  |
| Row 3 | a[2][0]  | a[2][1]  | a[2][2]  | a[2][3]  |

|       | Column 1     | Column 2     | Column 3     | Column 4     |
|-------|--------------|--------------|--------------|--------------|
| Row 1 | 1<br>a[0][0] | 2<br>a[0][1] | 3<br>a[0][2] |              |
| Row 2 | 4<br>a[1][0] | 5<br>a[1][1] | 6<br>a[1][2] | 9<br>a[1][3] |
| Row 3 | 7<br>a[2][0] |              |              |              |

|   |    |   |
|---|----|---|
| 9 | -4 | 3 |
| 5 | 6  | 1 |
| 2 | 7  | 4 |

**-4 at (0,1)**

|    |   |   |
|----|---|---|
| -4 | 9 | 3 |
| 5  | 6 | 1 |
| 2  | 7 | 4 |

**1 at (1,2)**

|    |   |   |
|----|---|---|
| -4 | 1 | 3 |
| 5  | 6 | 9 |
| 2  | 7 | 4 |

**2 at (2,0)**



FAZ ER

# O programa JogodaVelha.JAVA:

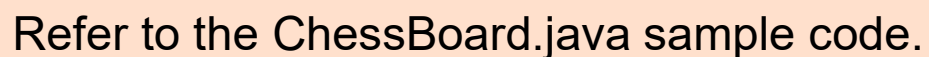
1. Implementar o jogo em JAVA.
2. Usar o scanner.

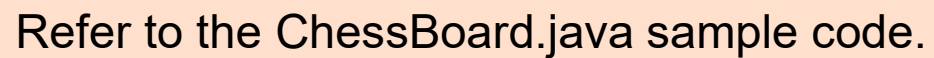
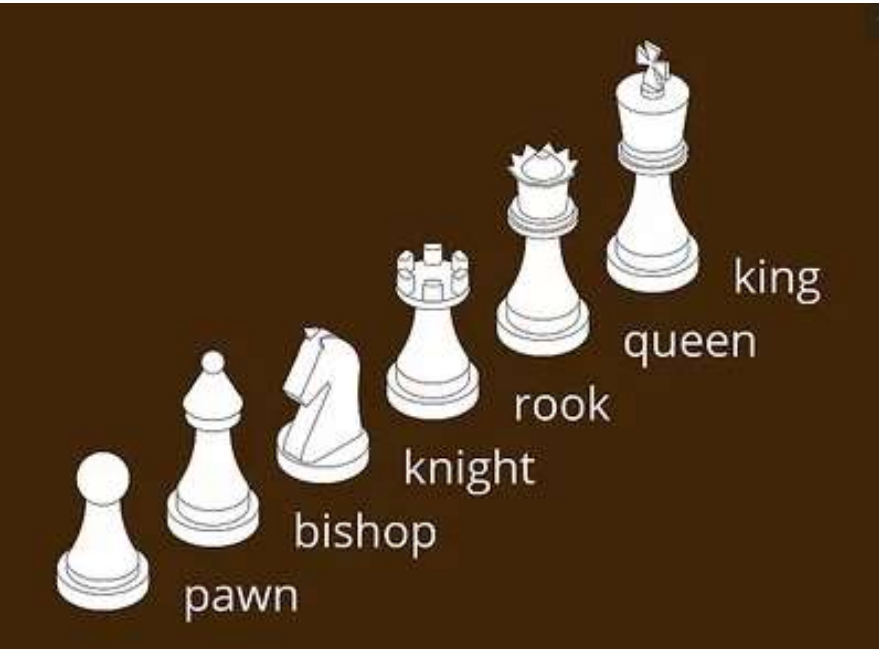
FAZER

- Criar um array que simule um tabuleiro de xadrez;
- Simular a movimentação de algumas peças, pelo tabuleiro;
- Validar a entrada de dados;

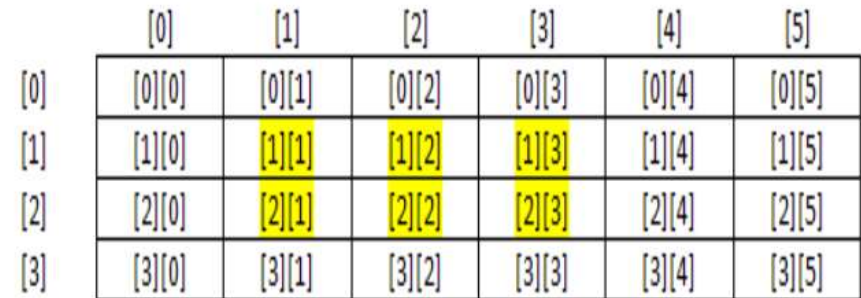
VM1066:1

► Array(8)





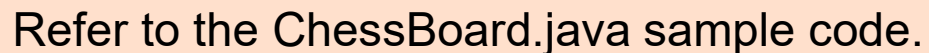
```
board[x][y] origem <- apagar;
```



Refer to the ChessBoard.java sample code.

[illegible]

- [illegible]



[illegible]

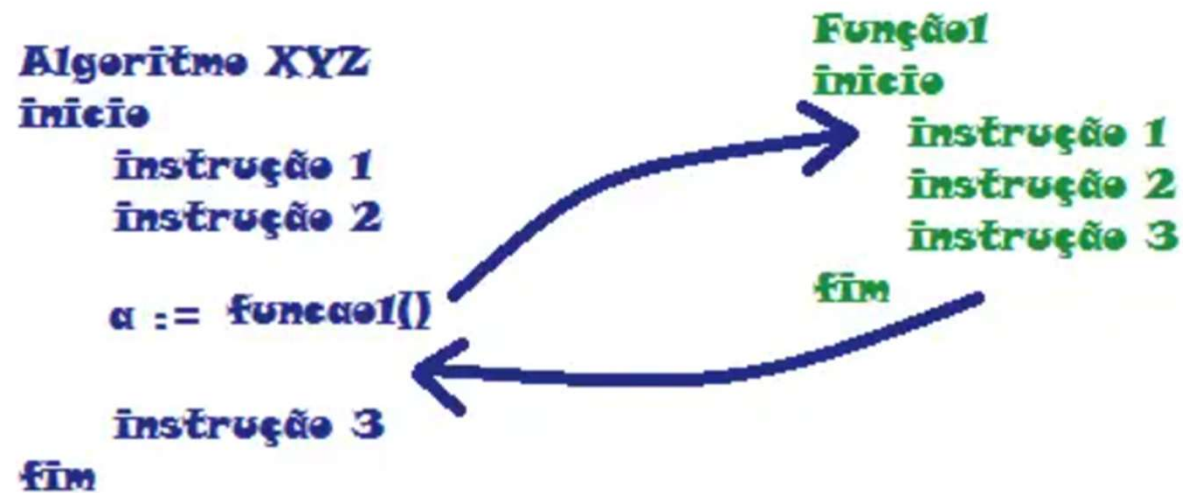
61

[illegible]

- Copyright © 2023 Accenture All Rights Reserved.

# Métodos

## Lógica de Programação





[illegible]

Diagrama de uma calculadora com botões de operação matemática:

- SOMA
- SUBTRAÇÃO
- MULTIPLICAÇÃO
- DIVISÃO

Calcule a expressão:  $10 + 20 - 30 \times 40 \div 50$

No desenvolvimento do código da calculadora vamos precisar criar as 4 funções de acordo com as operações definidas.

# Métodos

```
10
11 public static void main(String[] args) {
12
13 int x = -11;
14 int y = 5;
15
16 int[] nums = { 1, 2, 3, 4 };
17
18 System.out.println("Addition - " + add(x, y));
19 System.out.println("Subtraction - " + subtract(x, y));
20 System.out.println("Multiply - " + multiply(nums));
21 x = 3;
22 y = 0;
23 System.out.println("Divide - " + divide(x, y));
24
25 }

30 private static int add(int x, int y) {
31 int sum = x + y;
32 return sum;
33 }
34
35 private static int subtract(int x, int y) {
36 int diff = 0;
37 if (x > y) {
38 // complete the code
39 diff = x - y;
40 } else {
41 // complete the code
42 diff = y - x;
43 }
44 return diff;
45 }
```

The diagram illustrates the flow of execution between methods. A green circle highlights the opening curly brace of the `main` method. A brown arrow points from the `add(x, y)` call on line 18 to the `add` method definition starting at line 30. Another brown arrow points from the `subtract(x, y)` call on line 19 to the `subtract` method definition starting at line 35. A green arrow points from the `multiply(nums)` call on line 20 to the `multiply` method definition starting at line 40. The closing curly brace of the `main` method on line 25 is also highlighted with a green circle.

[illegible]

- 
- Diagrama de um método Java com comentários explicando suas partes:
- ```

11
12
13
14 public double calculoSoma(double a, double b) {
15
16     return a+b;
17
18 }
19

```
- Comentários e partes do código:
- Visibilidade do Método:** `public`
 - Tipo de Retorno:** `double`
 - Nome do Método:** `calculoSoma`
 - Parâmetro:** `(double a, double b)`
 - Início do corpo do método:** `{`
 - Assinatura do método:** `public double calculoSoma(double a, double b)`
 - Corpo do Método:** `return a+b;`
 - Fim do corpo do método:** `}`

greet()

Nenhum parâmetro com um tipo de retorno "nulo". Isso significa que o método não retorna nenhum valor;

```
package sef.module3.sample;
```

```
public class MethodSample {
```

```
public void greet(){
    System.out.println("Hello!");
}
```

greet(String)

- Você pode passar parâmetros para um método. Essas são consideradas variáveis locais
- Tem o mesmo nome que o método anterior, mas parâmetros diferentes
- Observe um modificador "estático". Isso significa que esse método é um método de classe e pode ser chamado sem uma referência a objeto

```
public static void greet(String name){
    System.out.println("Hello " + name +
"!");
}
```

```
public int sum(int x, int y) {  
    return x + y;  
}
```

thirdMethod

This method has a *int* return type. This requires the method to have a 'return' statement that returns an integer value.

- Copyright © 2023 Accenture All Rights Reserved. 68

- Copyright © 2023 Accenture All Rights Reserved.

214869676820706572666f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820

```
public class MethodSample {
```

Chame um método de instância através de seu objeto

Chame métodos de classe estaticamente e passe parâmetros

Chame um método de instância que aceite parâmetros e retorne valores

70

[illegible]

É uma coleção de uma ou mais instruções que executam uma tarefa específica

É uma sequência de objetos ou primitivos, todos do mesmo tipo



[illegible]

-
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a marker in her right hand. Two men in business suits are seated at a glass table in the foreground, looking towards the whiteboard. The man on the left is holding a folder, and the man on the right has his hand to his chin in a thoughtful pose. The office has large windows in the background, letting in bright light.

[illegible]

-
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a marker in her right hand. Two men in business suits are seated at a glass table in the foreground, looking towards the whiteboard. The man on the left is holding a folder, and the man on the right is resting his chin on his hand. The office has large windows in the background, letting in bright light.

[illegible]

-

