



# Application Delivery Fundamentals: Java

## Java Excellence

**accenture**>technology

Module 3: Introduction to Java – Parte 2

[illegible]

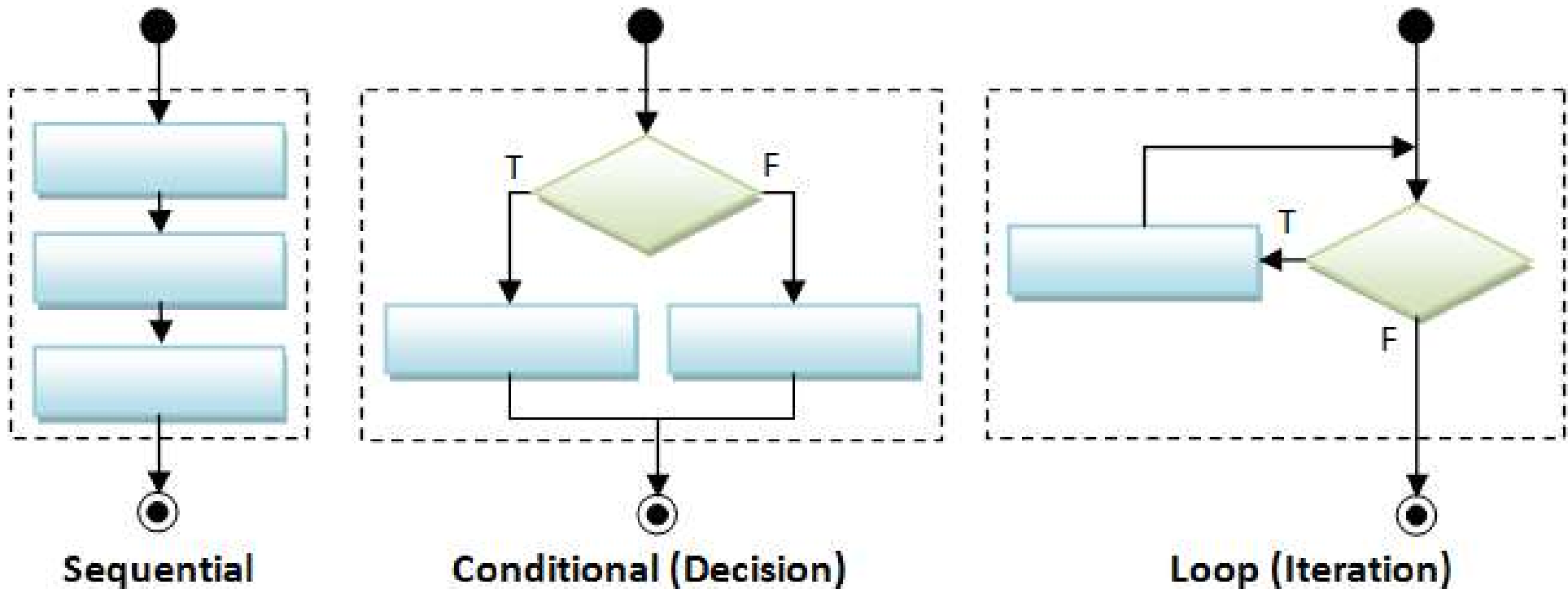
-

-

[illegible]

1

Um '**fluxo**' de programas refere-se à ordem em que as instruções são executadas;





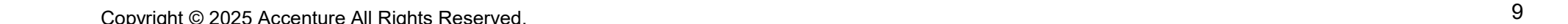
# Types of Flow Control

- *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;

[illegible]

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



[illegible]

[illegible]

[illegible]

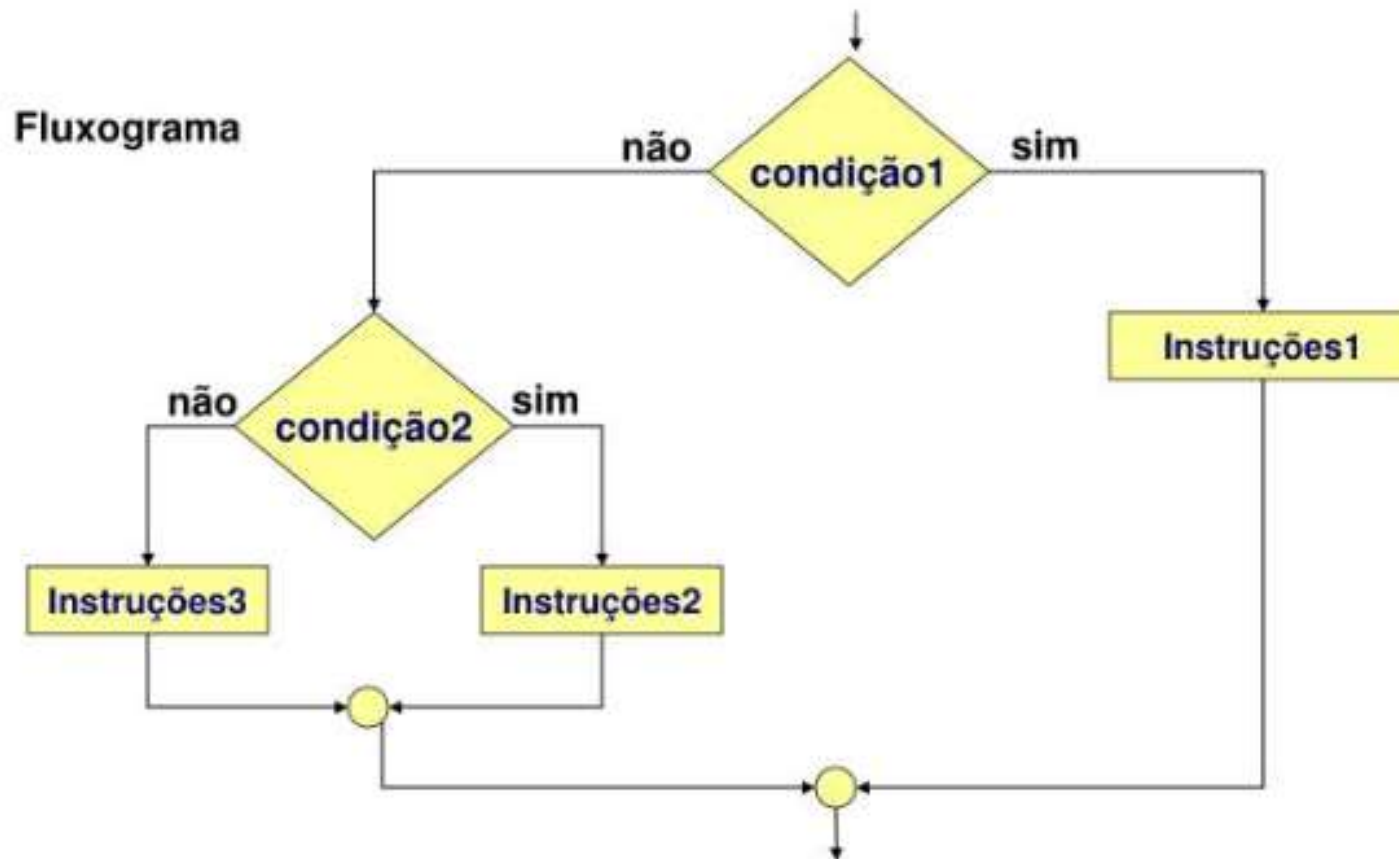
- ```
Syntax:  if ( condition ) {  
           // statement(s) to be executed If condition is met  
        } else { // ( else statement is optional )  
           // statement(s) to be executed If condition is NOT met  
        }  
  
Example: int x = 1;  
        if ( x > 0 ) {  
            System.out.println(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



-

[illegible]

- Aninhamento



- 
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a pen in her right hand and a folder in her left. 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 laptop, and the man on the right is resting his chin on his hand. The setting is a modern office with large windows in the background.

**Syntax:** `switch (exp) {`

**Example:** `int x = 1;`



[illegible]

- 
- ```
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]

[illegible]

-

[illegible]

21

Refer to the ScannerExample.java sample code.

# Tipos de input

- A classe scanner esta presente no pacote **java.util**;

| 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 from an elevated perspective. The green football pitch is centrally located with white markings. The stands are filled with red seating. A large, curved white roof structure covers the left side of the stadium. A flag is visible on a pole at the top of the stands.

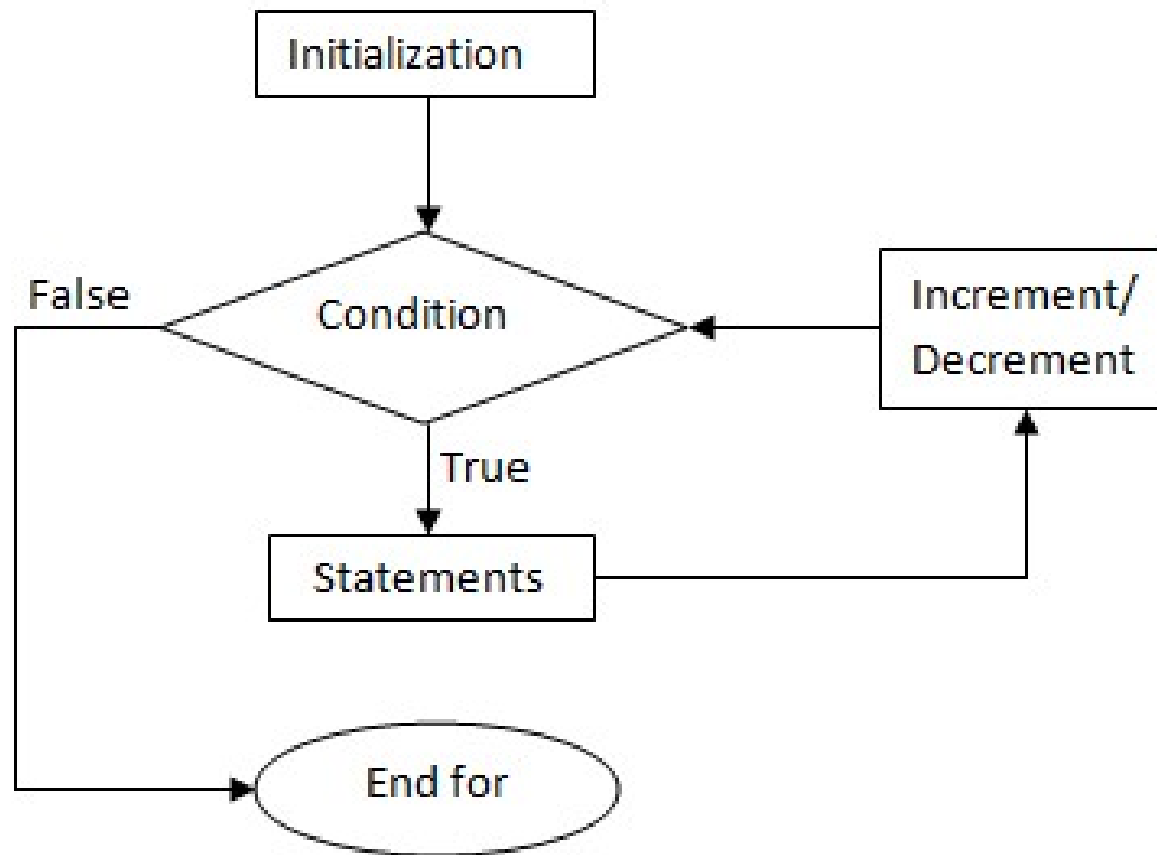


# Types of Flow Control

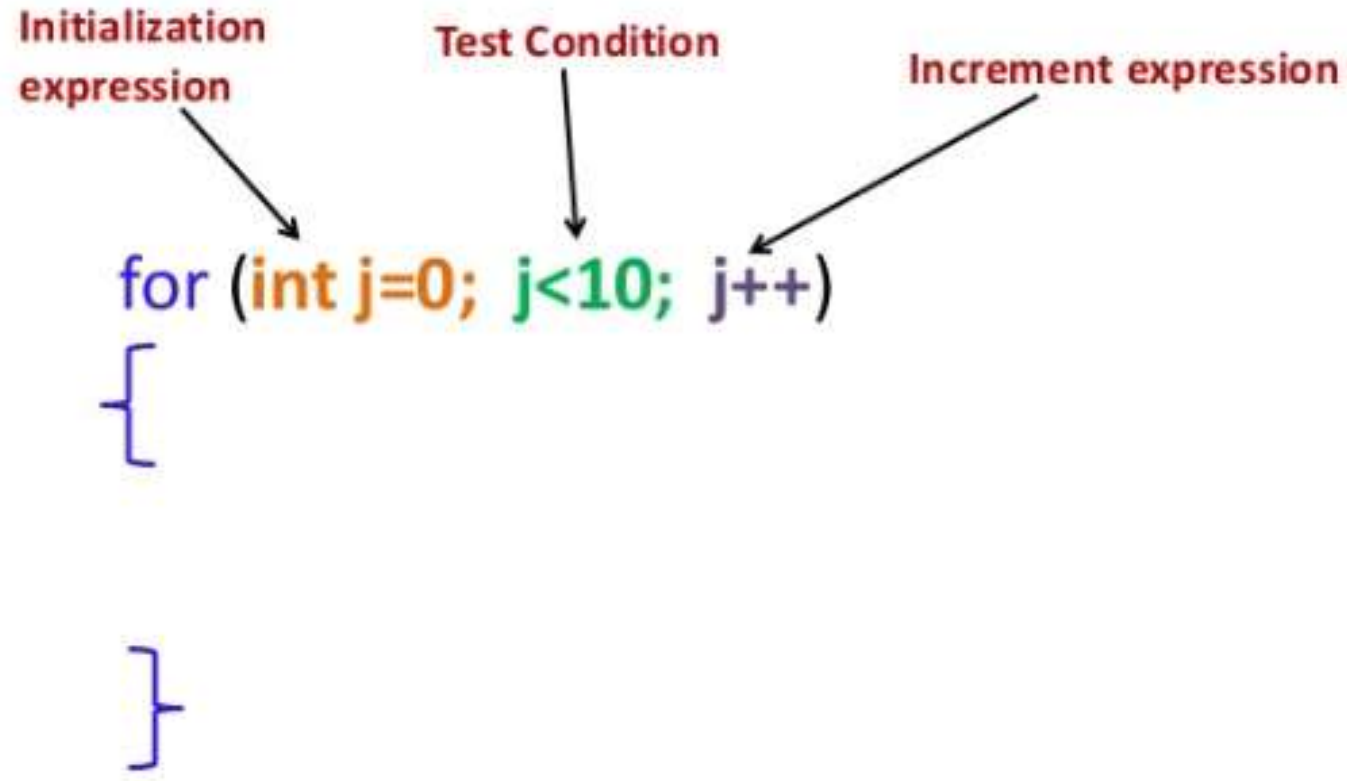
- Estruturas de iteração executam instruções repetidamente com base em uma condição
- Esse tipo de controle de fluxo pode ser implementado usando as diferentes instruções de loop:
  - for-loop
  - while-loop
  - do-while loop

# For Loop

- Um loop for executa instruções repetidamente se uma determinada condição for atendida



- Syntaxe



O **primeiro argumento** do for é uma variável para auxiliar a controlar a quantidade de repetições a serem executadas. Essa variável é chamada de variável de controle.

O **segundo argumento** do for é utilizado para definir até quando o for será executado.

O **terceiro argumento** indica o quanto a variável de controle será modificada no final de cada execução dentro do for.

[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.

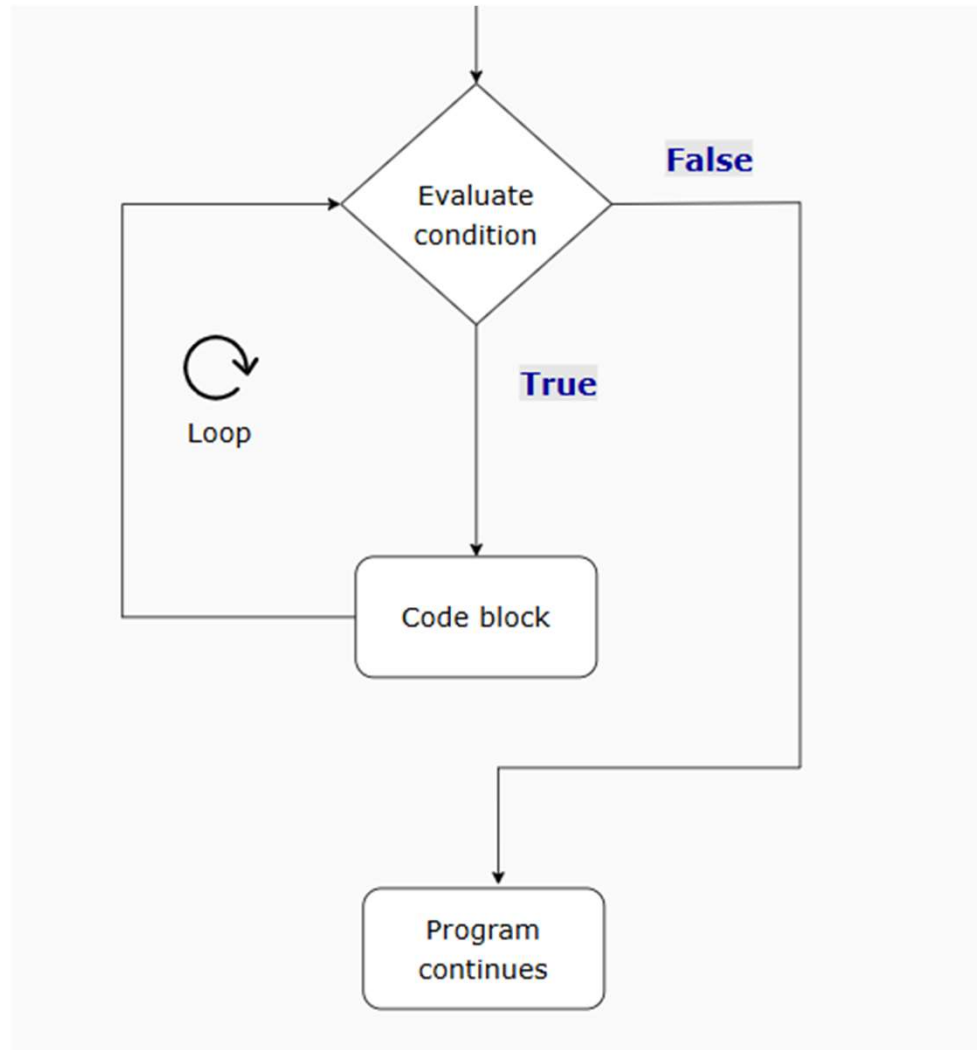
[illegible]

- 
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a pen 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.

Copyright © 2025 Accenture All Rights Reserved.

# While Loop

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



```
while (Boolean expression)
{
    // Operators
}
```



Refer to the WhileLoopSample.java sample code.

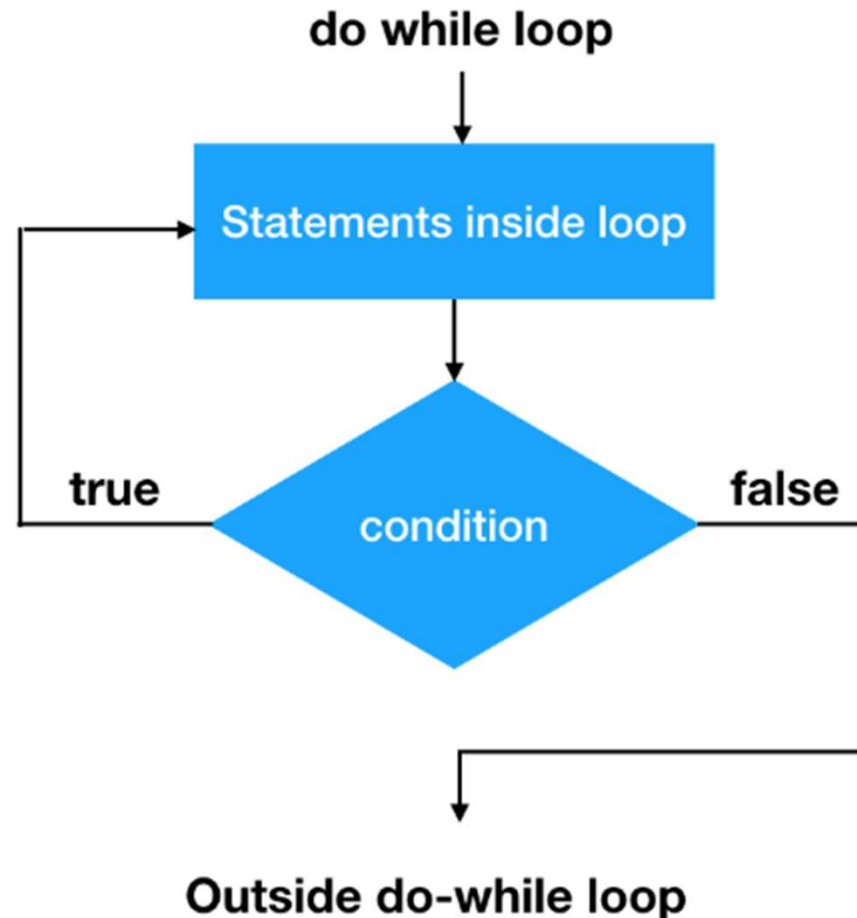


[illegible]

-

# Do-While Loop

- `do-while()` executa instruções repetidamente (pelo menos uma vez) enquanto a condição permanece verdadeira;





# JavaScript Controle de Fluxo

**While vs do .. while**



- ```
do
{
    // Operators
} while (Boolean expression);
```



[illegible]

**uma vez? SIM**

# elementos do array? **SIM**

# Tipos de controle de Fluxo?

- **for-loop**
- **while-loop**
- **do-while loop**
- **If - else**
- **Switch**

[illegible]

Copyright © 2025 Accenture All Rights Reserved. 39

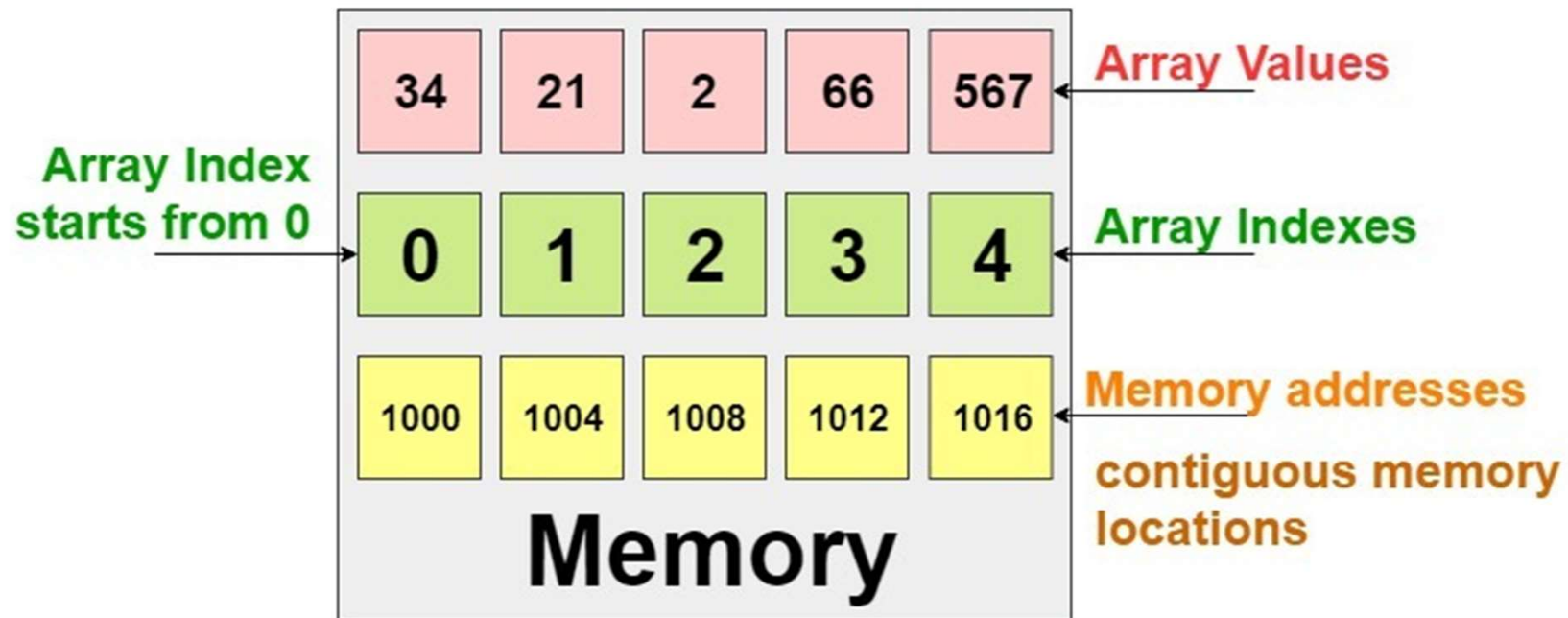




[illegible]

- Sintaxe Básica

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



[illegible]

[illegible]

- ## Como criamos um array vazio?




0 0 0 0 0

42 51 63 90 87



[illegible]

- NEW



100

[illegible]

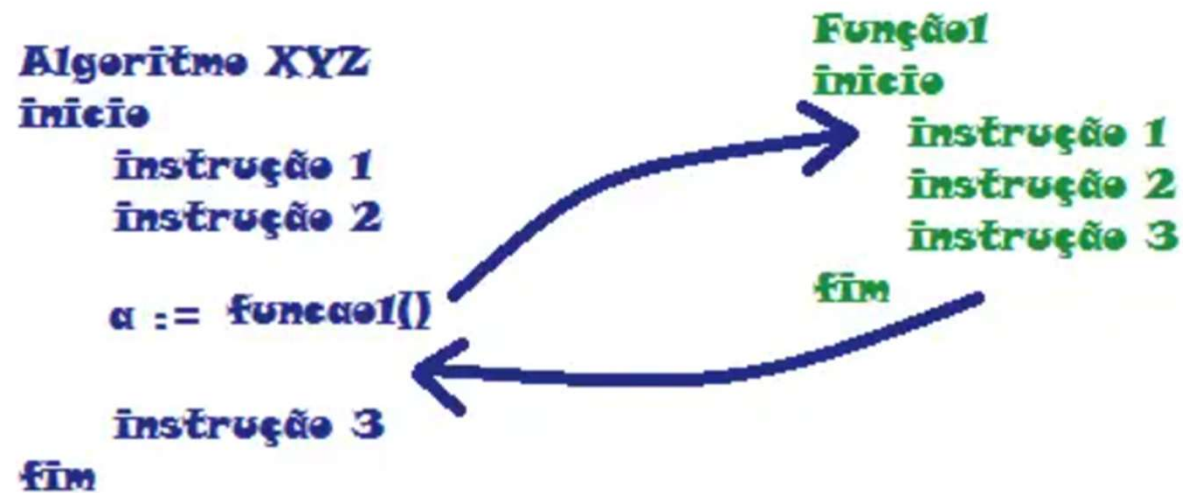
46

[illegible]

- Copyright © 2025 Accenture All Rights Reserved.

# Métodos

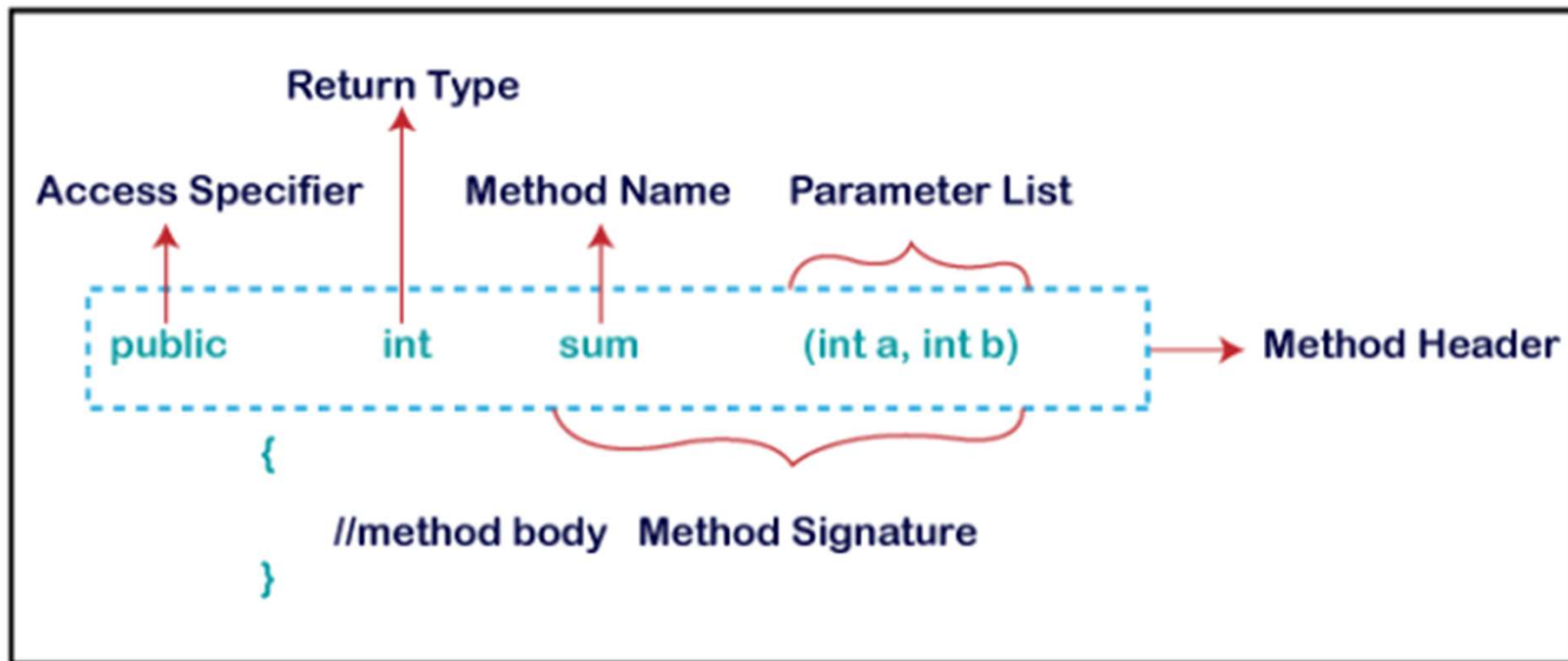
## Lógica de Programação





# Métodos

- Sintaxe básica de uma declaração de método Java:
- A assinatura de um método é uma combinação do nome e dos parâmetros de um método que identificam exclusivamente um método;



# Métodos

## Return

```
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 return values from the `add` and `subtract` methods to the `main` method. A brown arrow originates from the `return sum;` statement in the `add` method (line 32) and points to the `add(x, y)` call in the `main` method (line 18). A green arrow originates from the `return diff;` statement in the `subtract` method (line 44) and points to the `subtract(x, y)` call in the `main` method (line 19). Another green arrow points from the `multiply(nums)` call in the `main` method (line 20) to the `multiply` method definition, which is partially obscured. The `main` method's opening curly brace is circled in green, and its closing curly brace is also circled in green.

[illegible]

# “Argumentos”

11

```
function add(x, y) {  
    return x + y;  
}
```

14

argumentos

Argumentos são informações que podem ser passadas para um método quando ele é chamado.

## 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.

```
public class MethodSample {
```

## Chame um método de instância através de seu objeto

## Chame métodos de classe estaticamente e passe parâmetros

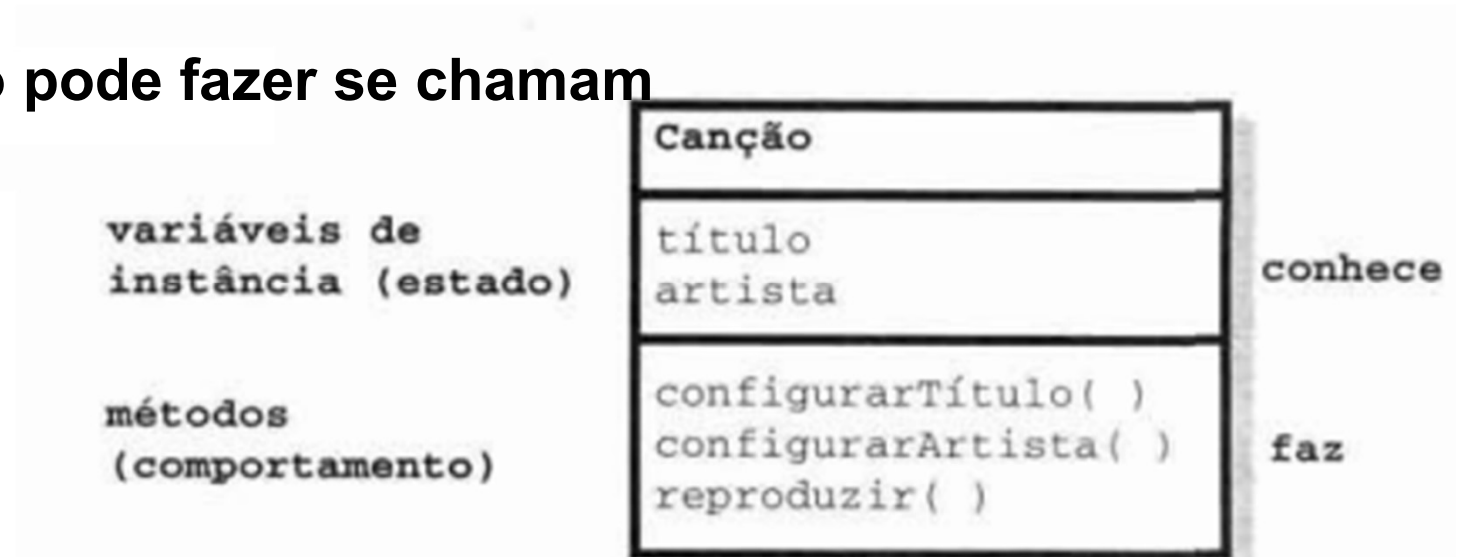
Chame um método de instância que aceite parâmetros e retorne valores

} }

[illegible]

- variaveis de instância.

## - métodos



[illegible]

Diagram illustrating the four basic arithmetic operations available on a calculator:

- SOMA (Addition)
- SUBTRAÇÃO (Subtraction)
- MULTIPLICAÇÃO (Multiplication)
- DIVISÃO (Division)

The central label is CALCULADORA.

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

Refer to the `Calculator.java` e `NumToWordsUsingMethod.java` sample code.





- 
- A woman in a dark business suit is standing and writing on a whiteboard. She is holding a pen in her right hand and a folder in her left. 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, and the man on the right is resting his chin on his hand. The room 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 pen in her right hand and a folder in her left. 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 laptop, and the man on the right has his hand on his chin, appearing to be in deep thought. 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 pen and looking at the board. 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 room has large windows in the background, showing a city view.

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

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

## Valor passado para uma método.

[illegible]

Copyright © 2025 Accenture All Rights Reserved. 61

[illegible]

# “VarArgs”

# NomeMétodo(Tipo...argumento )

Permitem que NENHUM argumento seja passado, bem como vários argumentos sejam passados quando o método é chamado.



Refer to the varArg1.java sample code.

[illegible]

```
public class varArg2 {
```

}



Refer to the varArg2.java sample code.

[illegible]

-



[illegible]

# Variáveis: Escopo

- Refere-se a partes ou seções de um programa em que a variável tem valor e é considerada 'visible';

## Tipos de variáveis por escopo:

# Class Variables

Compartilhado por todas as instâncias de uma classe. Identificado pela palavra-chave **static**.

# Instance Variables

Pertencer a uma instância de uma classe. São exclusivos para cada instância;

# Local Variables

São acessíveis apenas dentro de sua localidade e geralmente declarados dentro de um método;

## instance method

- ```
1: class VerificaEscopo{
2:     int escopoA;
3:     public void metodo(int escopoB) {
4:         int escopoC;
5:     }
6:
7: }
```

# Variáveis: Escopo

## Class Variable

É uma variável cujo valor é comum a todos os objetos membros da classe.

## Local Variable

Declarados dentro de métodos e / ou sub-rotinas. *aString* é local no método *main*

## Instance Variables

Possui um valor diferente para cada objeto instanciado.

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

```
public class VariableScope {
```

```
    /**
```

```
     * @param args
```

```
     */
```

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

```
        String aString = "This is a local  
        variable";
```

```
    }
```

```
}
```

```
class Employee {
```

```
    public static int totalCount = 0;
```

```
    private String myFirstName;
```

```
    private String myLastName;
```

```
    private int myAge;
```

```
}
```

214869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820

## O Código abaixo esta correto?

```
package unicesumar.ead.programacao;

public class Pessoa {

    private String nome;
    private int idade;

    private String falar(){

        String frase = "Olá";
        return frase;
    }

    public void andar(){
        System.out.println(nome + frase);
    }

}
```



## O programa LocalizaNúmero.JAVA:

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

Copyright © 2025 Accenture All Rights Reserved.

[illegible]



# Variáveis: Escopo

## Shadowing

Variáveis globais ficam ocultas em trechos do Código que são escopo de outras variáveis com o mesmo nome;

```
public class Parede {  
    private String cor;  
  
    public Parede() {  
        this.cor = "branca";  
    }  
  
    public void colorir(String cor) {  
        cor = "verde";  
    }  
}
```

O que acontece com o atributo de classe, ao executar o método colorir?

# Questions and Comments

- What questions or comments do you have?

