

Module 3: Introduction to Java - Parte 2

Module Objectives

- No final deste módulo, os participantes poderão:
 - Identifique e crie código fonte estruturado corretamente;
 - Articular a importância do método "principal";
 - Identifique e crie declarações e expressões apropriadas;
 - Descreva o conceito de variáveis;
 - Identifique as regras para o escopo das variáveis;
 - Descreva os tipos de dados primitivos e seus intervalos;
 - Identifique as regras para atribuir valores a variáveis;
 - Descrever o conceito de matrizes;
 - Identifique operadores Java e explique seus diferentes usos;



Module Objectives (cont.)

- No final deste módulo, os participantes poderão:
 - Explicar o conceito de controle do 'fluxo do programa';
 - Controlar 'fluxo de programa' usando as diferentes instruções de fluxo de controle;
 - Descrever o conceito de métodos;
 - Identifique e crie métodos estruturados corretamente;
 - Descreva o conceito de "chamada de método";

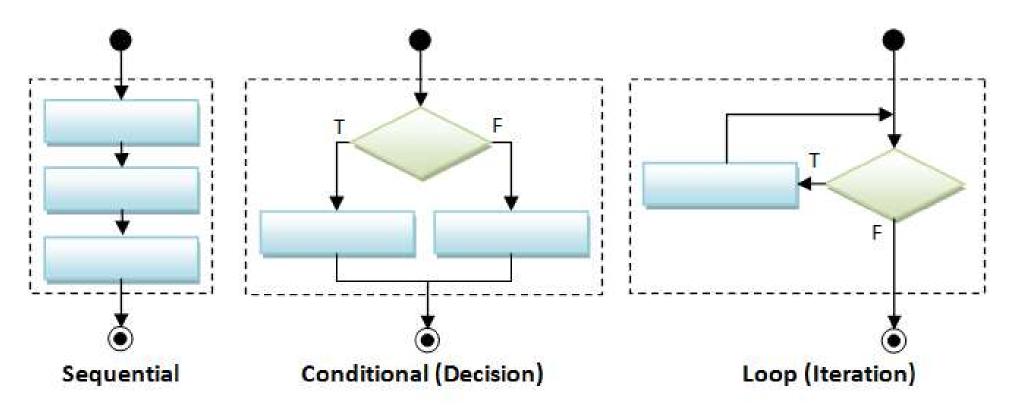


Flow Control

"Controle de Fluxo"

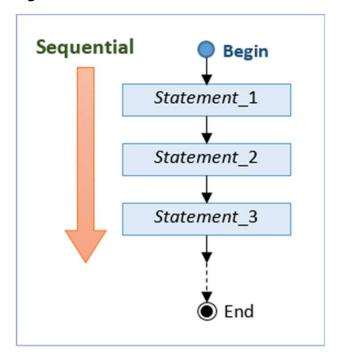
Types of Flow Control

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



Flow Control

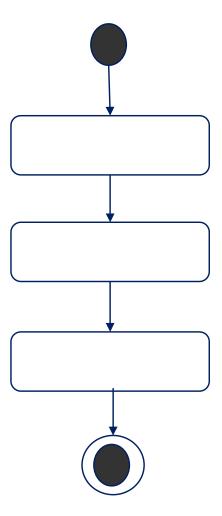
Por padrão, as instruções são executadas sequencialmente;



 As instruções de controle de fluxo são usadas para alterar ou modificar o caminho de execução das instruções quando certas condições estão presentes;

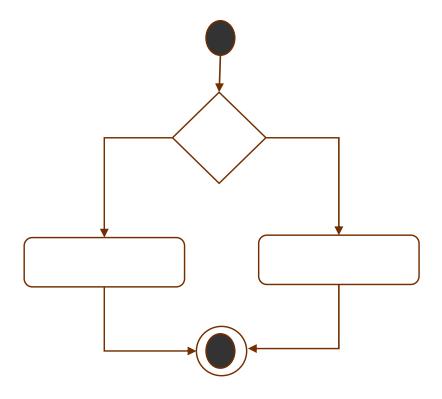
Types of Flow Control

- 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;

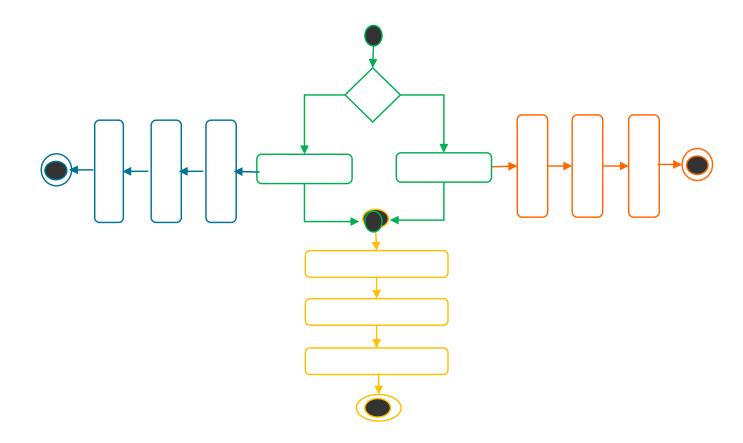


Types of Flow Control

- As estruturas de seleção executam uma determinada ramificação com base nos resultados de uma condição booleana;
- Esse fluxo é útil em cenários de tomada de decisão e pode ser implementado usando as instruções if-else ou switchcase;

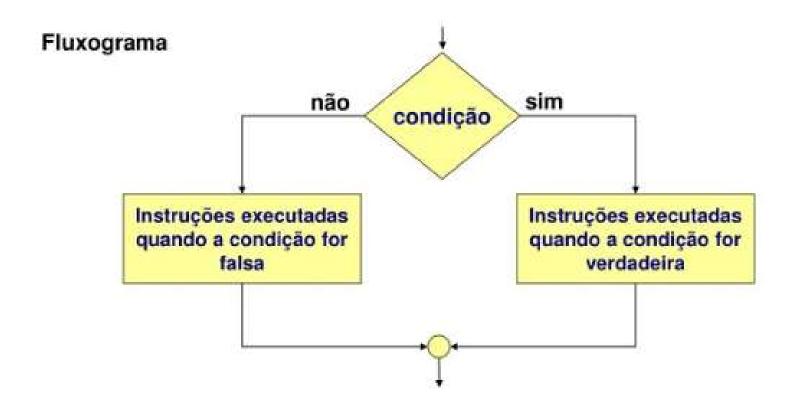


Flow Control



If-Else

 O resultado de uma instrução if-else executa é baseado em uma condição



If-Else

 O resultado de uma instrução if-else executa é baseado em uma condição

Operador Ternário



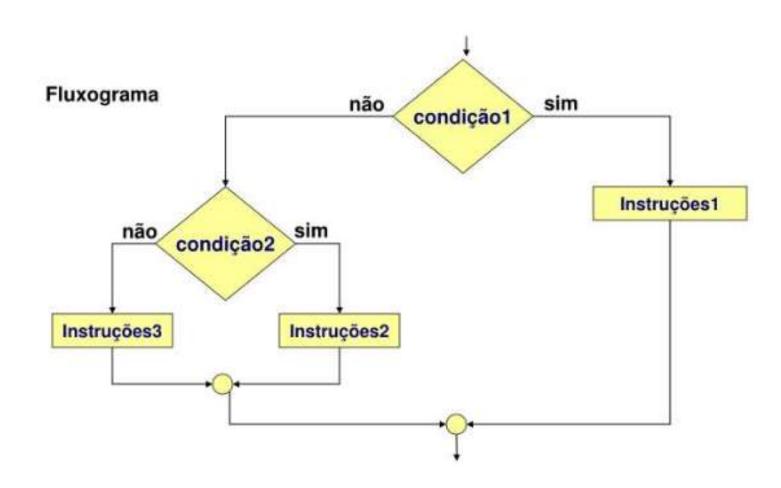
Activity

- Abra o arquivo 'FindLargest.java' do package sef.module3.activity
- Faça o seguinte:
 - Complete o código. Use a instrução if-else para encontrar o maior dos dois números dados;
 - Imprima o resultado;



If-Else

Aninhamento



Activity

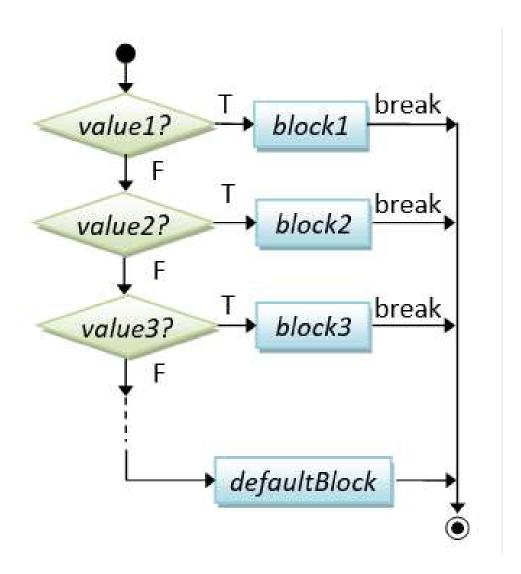
- Abra o arquivo 'FindLargest.java' do package sef.module3.activity
- Faça o seguinte:
 - Altere o programa para: Encontrar o maior de três números;
 - Leia os números do teclado;
 - Imprima o maior;



 Uma opção switch-case permite que o programa escolha quais instruções executar com base em uma condição

```
Syntax:
                   switch (exp) {
                   case val1: // statements here
                             break:
                   case val2: // statements here
                            break;
                   default: // statements here
Example:
                   int x = 1;
                   switch (x) {
                   case 1: System.out.println ("Value of x is 1");
                            break:
                   case 2: System.out.println ("Value of x is NOT 1");
                            break;
                   default: System.out.println ( "Value of x is NULL");
```

Break



```
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.

Break

Activity

- Abra o arquivo 'NumToWords.java' do package sef.module3.activity
- Faça o seguinte:
 - Complete o código e imprima o valor do texto do número 5



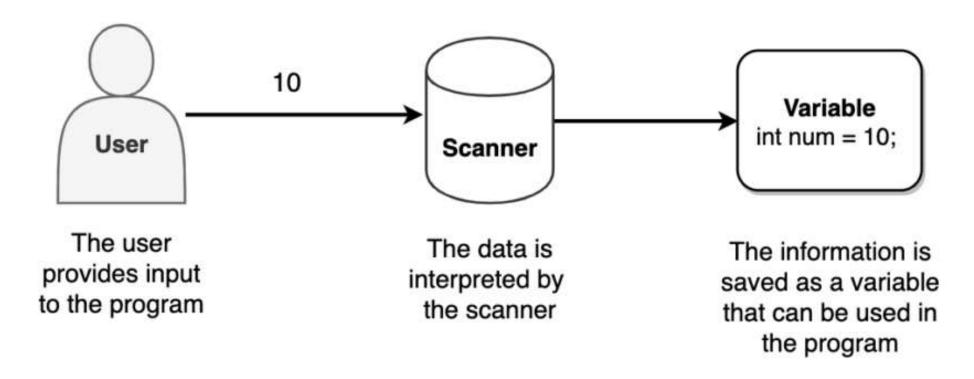
Scanners

"Scanner"

Scanners

Entrada de dados via teclado

 Você pode usar o Scanner para ler um número no teclado e atribuir seu valor a uma variável;



i

Refer to the ScannerExample.java sample code.

Scanner

Tipos de input

• A classe scanner esta presente no pacote java.util;

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





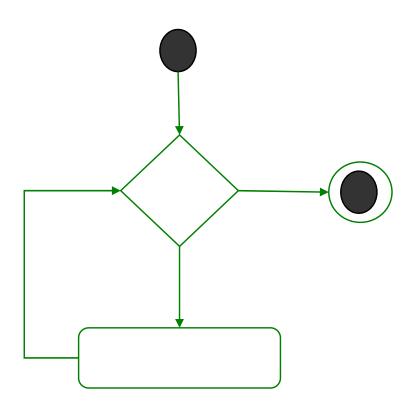
Atividade 1

- Um estádio oferece três categorias de assentos:
 - Classe A R\$ 50
 - □ Classe B R\$ 30
 - □ Classe C R\$ 20
- Escreva um programa que pergunta quantos bilhetes de cada classe foram vendidos, e, em seguida, exibe a renda gerada pela venda dos ingressos.



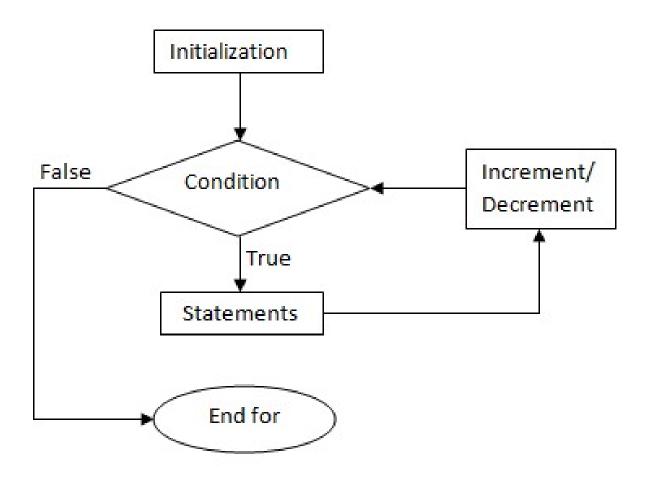
Types of Flow Control

- 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



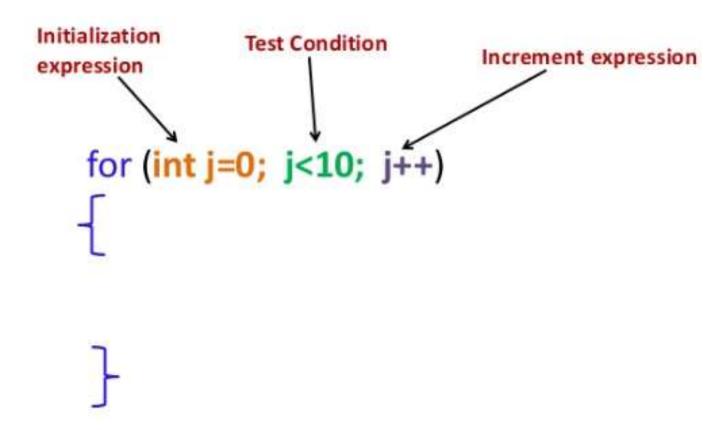
For Loop

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



For Loop

Sintaxe



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.

For Loop

 É usado para iterar apenas sobre a matriz. Nunca use para iterar sobre a coleção.

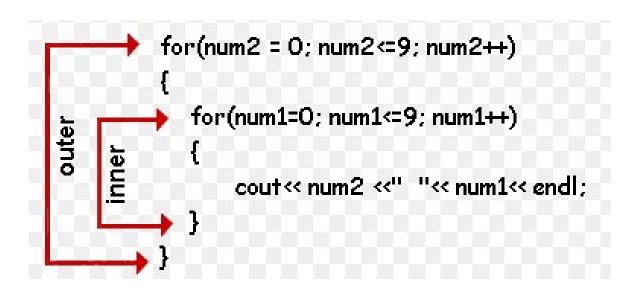
Activity

- Abra o arquivo 'AddWholeNum.java' do package sef.module3.activity
- Faça o seguinte:
 - Complete o código e escreva um loop for para soma todos os números inteiros de 50 a 100
 - Imprima o resultado



Activity

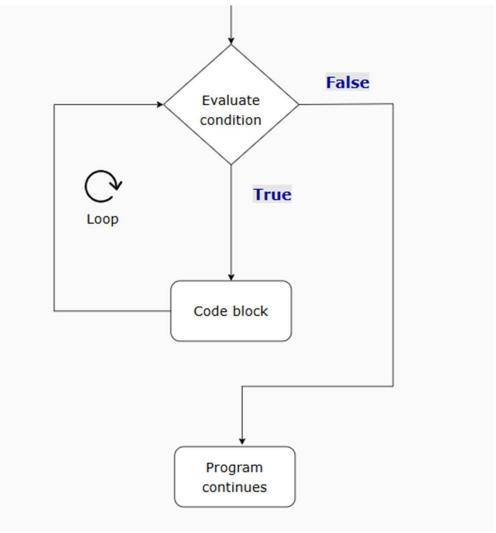
- Abra o arquivo 'MultiplicationTable.java' do package sef.module3.activity
- Faça o seguinte:
 - Complete o código e escreva loops para imprimir a tabuada de 11 a 20





While Loop

 while() executa instruções repetidamente enquanto uma condição permanece verdadeira



While Loop

 while() executa instruções repetidamente enquanto uma condição permanece verdadeira

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

i

Refer to the WhileLoopSample.java sample code.

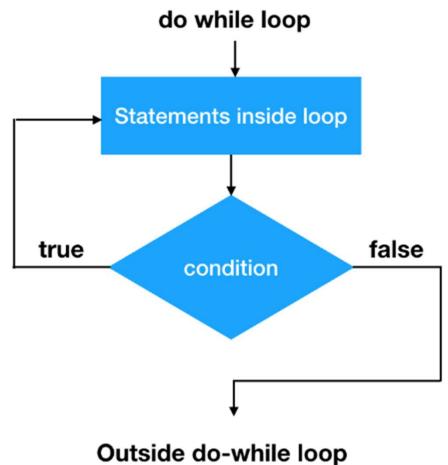
Activity

- Abra o arquivo 'PrintNumWithWhile.java' do package sef.module3.activity
- Faça o seguinte:
 - Complete o código e escreva um loop while para imprimir todos os números pares menores que 100



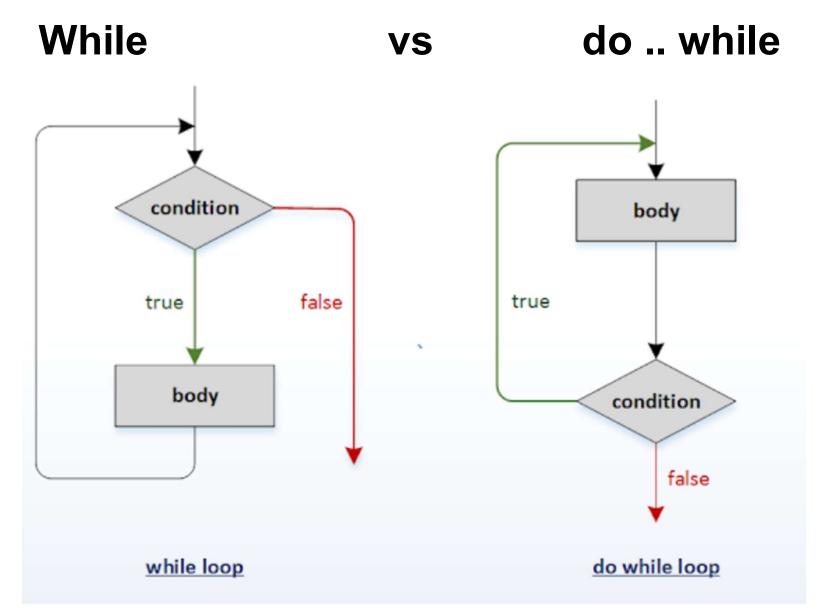
Do-While Loop

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





JavaScript Controle de Fluxo





JavaScript Controle de Fluxo

While do .. while **VS** while (not edge) { do (run(): run();) while (not edge):

Do-While Loop

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

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

i

Refer to the WhileLoopSample.java sample code.

Perguntas:

O do.... while é executado pelo menos uma vez? SIM

O for... each é executado até acabar os elementos do array? **SIM**

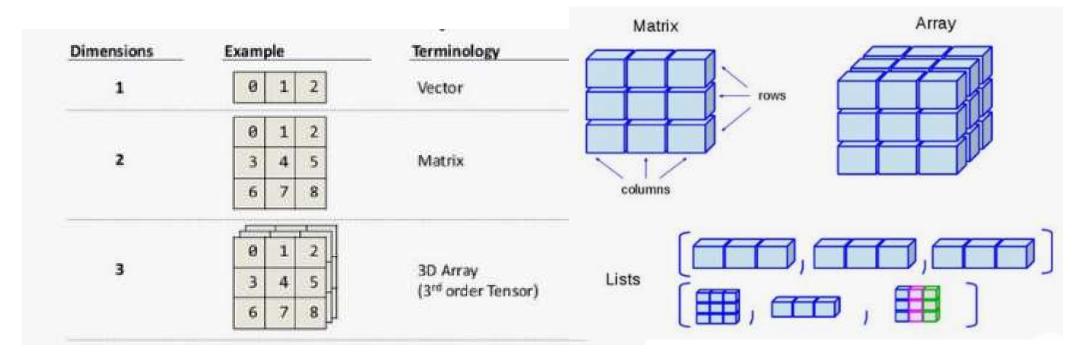
Tipos de controle de Fluxo?

- for-loop
- while-loop
- do-while loop

- If else
- Switch

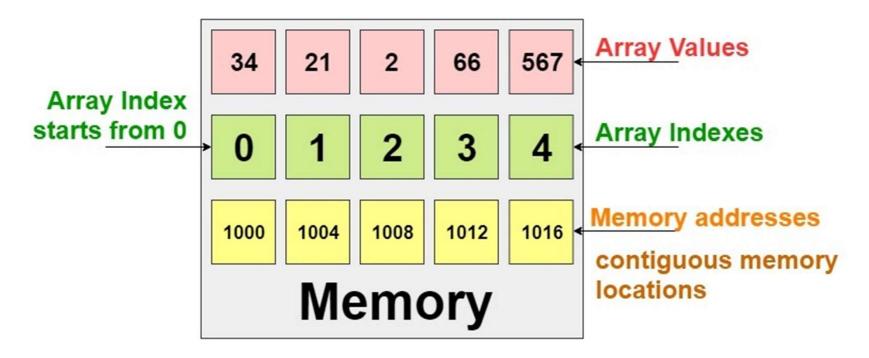
"Arrays"

 Um array é uma sequência de objetos ou primitivos, todos do mesmo tipo sob o nome de um identificador

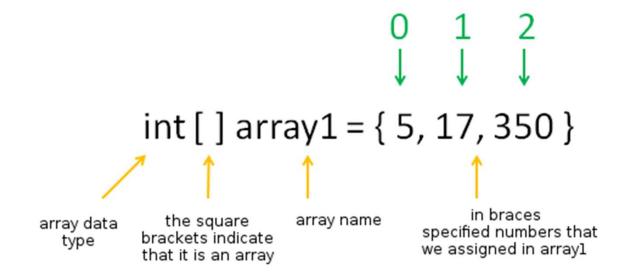


Sintaxe Básica

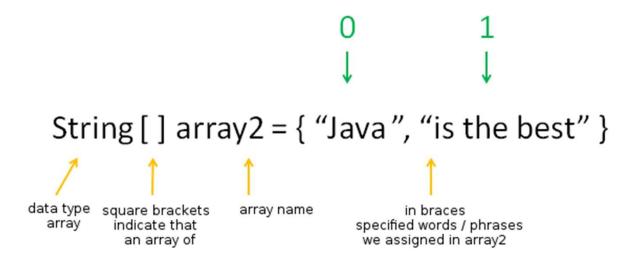
int $x[] = new int[] {34, 21, 2, 66, 567};$



Sintaxe básica:



Sintaxe básica:



Sintaxe básica:

Como criamos um array vazío?



```
int arr[] = \{42, 51, 63, 90, 87\};
```



Refer to the ListagemMes.java e LocalizaNumero.java sample code.

Sintaxe básica:

NEW

Como criamos um array vazío?

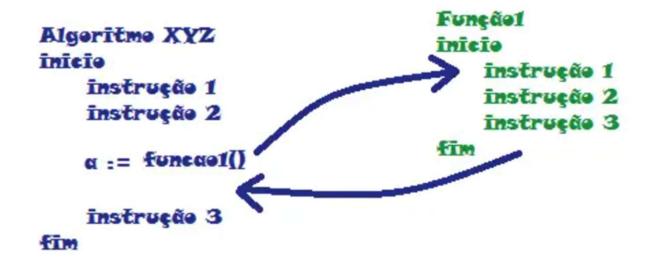
i

Refer to the NumerosInteiros1.java e NumerosInteiros2.java sample code.

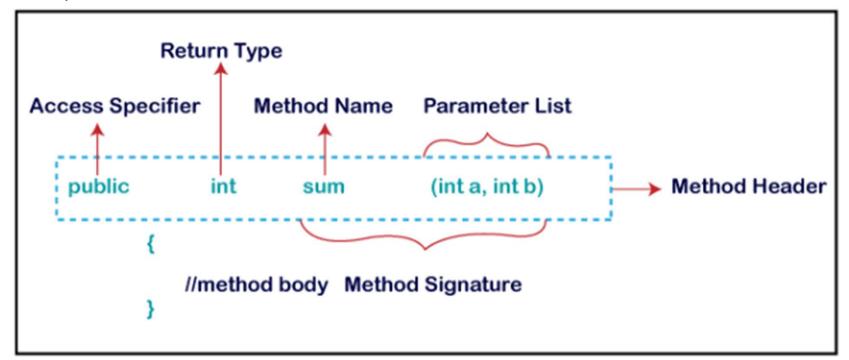
"Métodos"

- Um método é uma coleção de uma ou mais instruções que executam uma tarefa específica;
- Um método em Java é equivalente a uma função, subrotina ou procedimento em outras linguagens de programação;
- Métodos ajudam a modularizar a classe;
- Uma classe pode ter vários métodos;
- A classe fica mais legível;
- Promove a facilidade na manutenção;
- Melhoram a organização da classe;
- Promove o reuso;
- Previne duplicação de código e retrabalho;

Lógica de Programação



- 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;

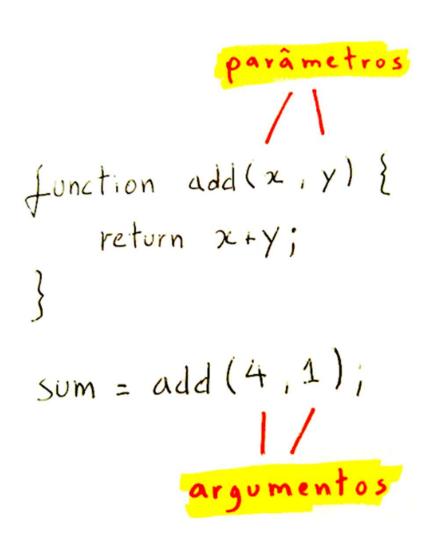


Return

```
private static int add(int x, int y) {
       public static void main(String[] args) {
11∘
                                                                        31
                                                                                   int sum = x + y;
                                                                        32
                                                                                   return sum;
13
           int x = -11;
                                                                       33
           int y = 5;
14
15
                                                                              private static int subtract(int x, int y) {
           int[] nums = { 1, 2, 3, 4 };
16
                                                                                   int diff = 0;
17
                                                                                   if (x > y) {
                                                                        37
           System.out.println("Addition - " + add(x, y));
                                                                       38
                                                                                      // complete the code
           System.out.println("Subtraction - " + subtract(x, \checkmark);
19
                                                                       39
                                                                                      diff = x - y;
           System.out.println("Multiply - " + multiply(nums))
20
                                                                       40
                                                                                   } else {
21
           x = 3;
                                                                                      // complete the code
           y = 0;
                                                                                      diff = y - x;
           System.out.println("Divide - " + divide(x, y));
23
                                                                       43
44
24
                                                                                   return diff;
```

"Argumentos"

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



Method Declaration

greet()

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

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

```
package sef.module3.sample;
public class MethodSample {
  public void greet(){
        System.out.println("Hello!");
  public static void greet(String name) {
       System.out.println("Hello " + name +
   w!");
  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.

Method Declaration

```
package sef.module3.sample;

public class MethodSample {

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

Chame um método de instância através de seu objeto

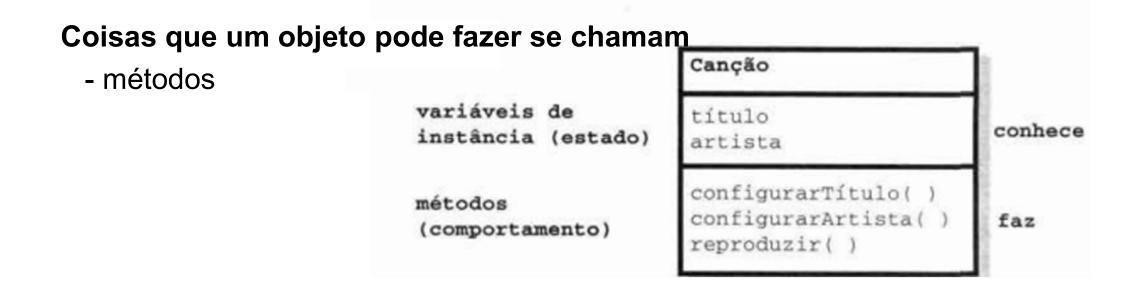
Chame métodos de classe estaticamente e passe parâmetros

Chame um método de instância que aceite parâmetros e retorne valores

```
public static void greet(String name) {
      System.out.println("Hello " + name + "!");
public int sum(int x, int y) {
      return x + y;
public static void main(String arg[]) {
      MethodSample sample = new MethodSample();
      sample.greet();
      greet("Java Teacher");
      MethodSample.greet("Java Student");
      System.out.println("Sum of 1 and 2 is " +
               sample.sum(1, 2));
```

Coisas que um objeto conhece sobre si mesmo se chamam

- variaveis de instância.



Em uma calculadora quais são as suas 4 principais operações?



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

Atividades



Atividade 2

O programa calculator.JAVA:

- 1. Criar um menu de opções.
- 2. Usar o scanner.
- 3. Usar switch case.

Atividade 3

- 1. Refatorar o Código da Eleição.java (switch case).
- 2. Refatorar o numtoWordsWithMetodos.java (switch case).
- 3. Refatorar o numerosInteiros2.java (operador ternário).

- Abra o arquivo 'FindInArray.java' do package sef.module3.activity
- Faça o seguinte:
 - Complete o código para encontrar o menor número na matriz fornecida.
 - Complete o código para encontrar o maior número na matriz fornecida



- Abra o arquivo 'Calculator.java' do package sef.module3.activity.
- Observe que os métodos add () e multiply () são chamados de dentro do método main ()
- Faça o seguinte:
 - Complete o código para os métodos add () e multiply ()
 - Imprima os resultados



- Abra o arquivo 'NumToWordsUsingMethod.java' do package sef.module3.activity
- Faça o seguinte:
 - Dê uma olhada em NumToWords.java, no qual escrevemos o código para converter números inteiros em seus valores de texto
 - NumToWordsUsingMethod.java executará a mesma função usando o método printWord ()
 - Complete o código;



Perguntas:

O que é um método?

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

O que é um array?

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

O que é um argumento?

Valor passado para uma método.

"VarArgs"

"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.

```
"VarArgs"
```

```
public class varArg2 {

   public static String tresPontos(String...i) {
        String frase = "";
        for (String s : i) {
            frase += s;
        }
        return frase;
    }

   public static void main(String[] args) {
        System.out.println("Resultado: " + tresPontos("Naquele ", "dia"));
   }
}
```

Refer to the varArg2.java sample code.

- Crie um método Java chamado concatenateStrings que aceite um número variável de argumentos String. Esse método deve concatenar todas as cadeias de caracteres de entrada e retornar uma cadeia de caracteres única resultante.
- Use scanner



"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;

```
4
 5
   package org.businessapptester.monitoring.tcpserver.protocol;
 8= /**
                                                  class definition
     * My documentation.
 9
10
    @SuppressWarnings("unus
    public class MyClass {
12
                                                     instance variables
13
14
       private int intValue;
15
       private String stringValue;
16
17
        // method declaration
       public void doSomething(int intValue, String stringValue){
18E
           this.intValue = intVal
19
20
           this.stringValue = stringVal
21
           // do something with the values
22
23
                                                      instance method
24
```

 O escopo de uma variável informa onde ela pode ser utilizada

Exemplo:

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

Class Variable

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

Local Variable

Declarados dentro de métodos e / ou subrotinas. *aString* é local no metodo *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;
```

Atividade

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);
```





Atividade 4

O programa LocalizaNúmero.JAVA:

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

```
1 package sef.module3.sample;
 2 //Faça um programa para pesquisar o valor 8 no vetor dado:
 3 //inteiro vetor[] = \{1, 3, 5, 8, 9, 10\}
 4 public class LocalizaNumero {
      // Corrigir este algoritmo.
       public static void main(String[] args) {
           int vetor[] = {1,2,3,4,5,6,7,8,9,10};
           int numero = 8;
           boolean achou = false;
           for (int i=0; i<10; i++) {
               if (vetor[i] == 8){
                   achou = true;
13
           if (achou) {
               System.out.println("Achei");
               System.out.printf("Na Posição %d está localizado do numero %d.", i, vetor[i]);
```

"Shadowing"

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?



