



High performance. Delivered.

Application Delivery Fundamentals: Java

Module 3: Language Fundamentals - 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";

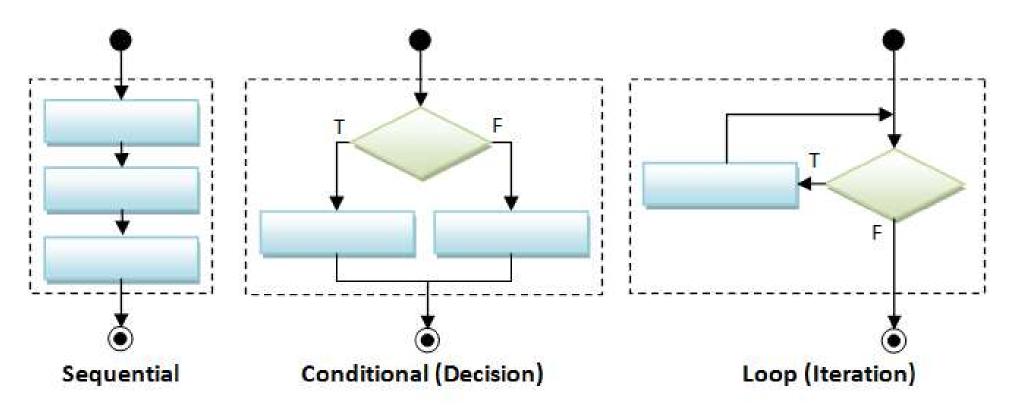


Flow Control

"Controle de Fluxo"

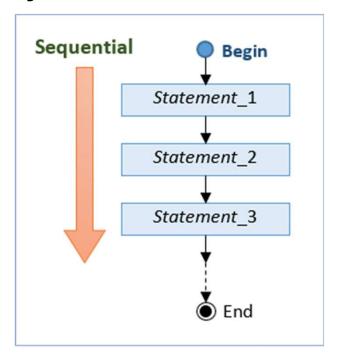
Types of Flow Control

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



Flow Control

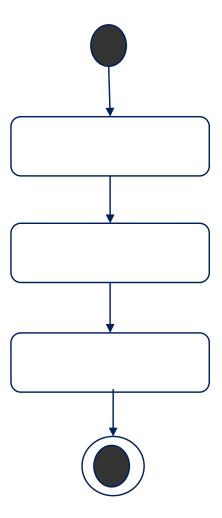
Por padrão, as instruções são executadas sequencialmente;



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

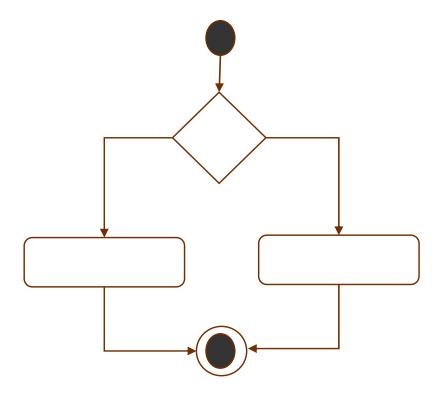
Types of Flow Control

- 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;

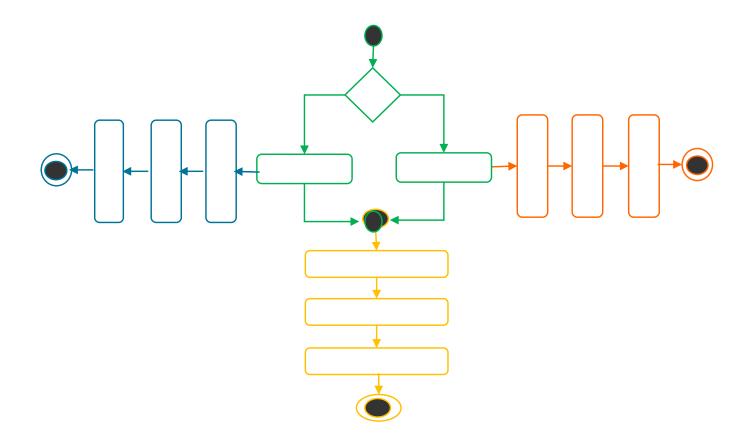


Types of Flow Control

- 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;



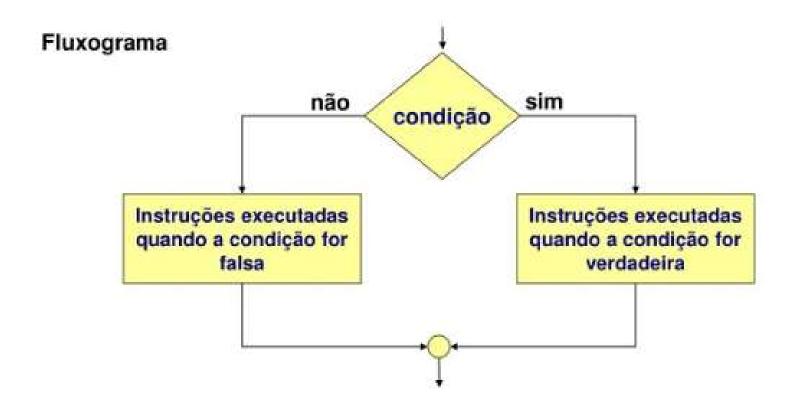
Flow Control



2f4869676820706572666f726d616c63652c2044656c6976657265642c27486967682070657266677266772677

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



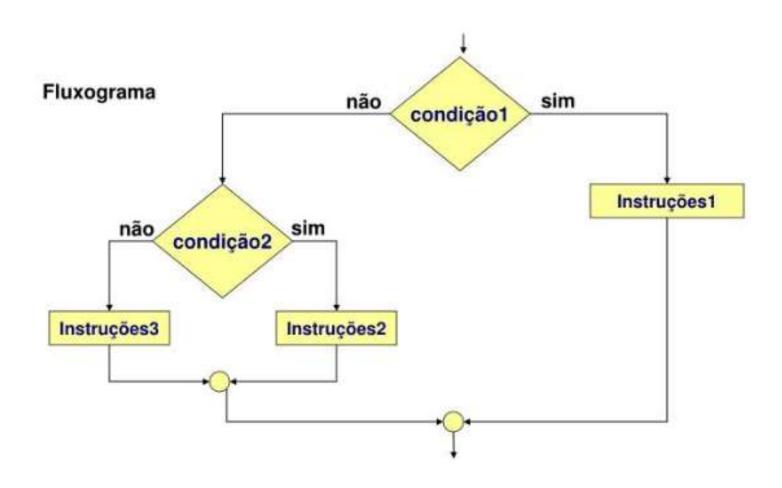
Activity

- 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

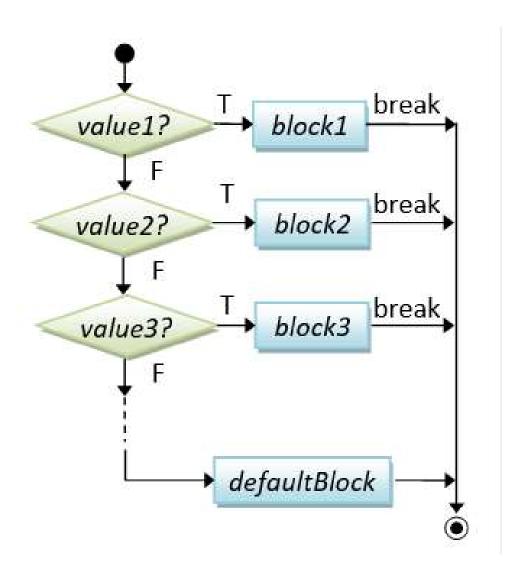
- Abra o arquivo 'FindLargest.java' do package sef.module3.activity
- Faça o seguinte:
 - Altere o programa para: Encontrar o maior de três números;
 - Leia os números do teclado;
 - Imprima o maior;



 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

```
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;
                System.out.println("Inside Case 5");
                break;
        default:
                System.out.println("Inside default case");
   System.out.println("After Switch Statement");
                                               nothernor of the property
```

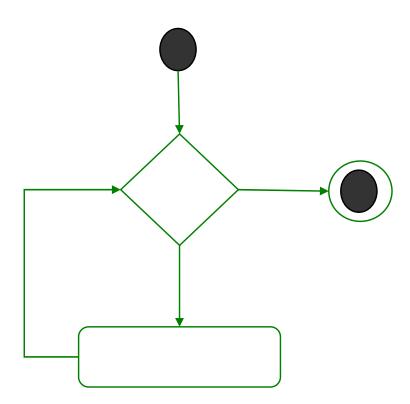
Activity

- 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



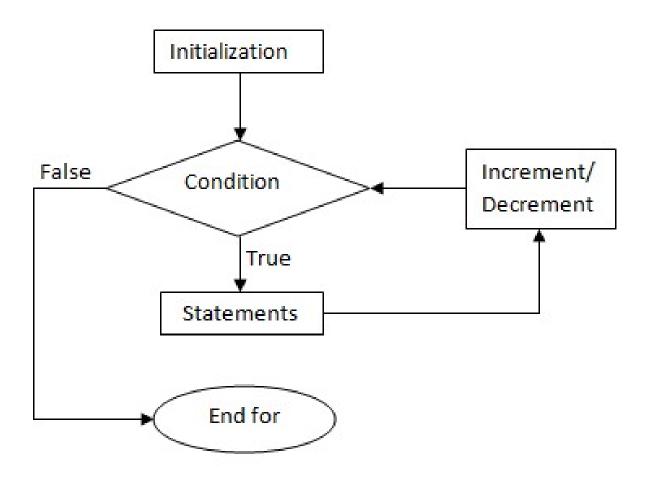
Types of Flow Control

- 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



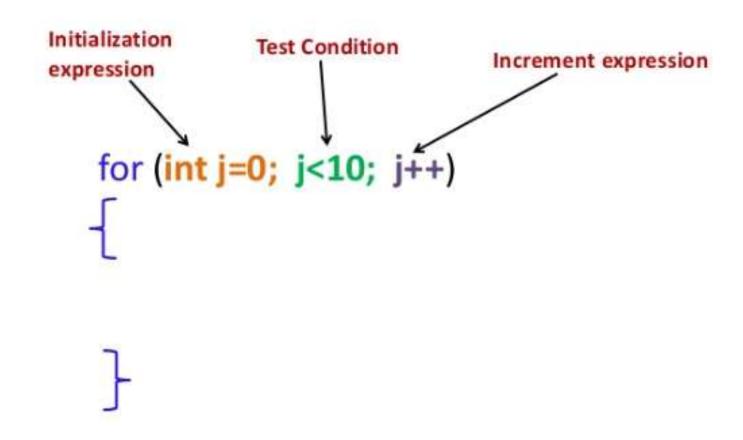
For Loop

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



For Loop

Sintaxe



For Loop

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

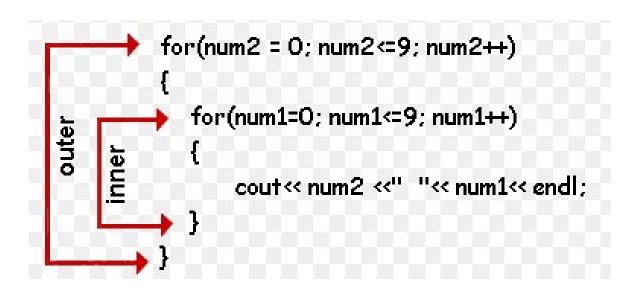
Activity

- 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





For-each Loop

- O loop for-each torna seus loops mais compactos e fáceis de ler.
- Quando ver os dois pontos (:), leia-os como "em" -> "para cada elemento da coleção".

For-each Loop

- O loop for-each torna seus loops mais compactos e fáceis de ler.
- Quando ver os dois pontos (:), leia-os como "em" -> "para cada elemento da coleção".

Activity For-each Loop

- Abra o arquivo 'AddWholeNum.java' do package sef.module3.activity
- Faça o seguinte:
 - Crie um objeto List <Integer>;
 - Usando um for tradicional, preencha o objeto de lista com 10 objetos Inteiros.
 - Usando um for-each, imprima os valores int dos objetos Integer.



Iterator<E> Loop

 Um Iterador <E> é um objeto que permite percorrer uma coleção e remover elementos da coleção seletivamente, se desejado. Você obtém um iterador para uma coleção chamando seu método iterador.

```
Syntax:
                    for ( Iterator<E> it = collection.iterator(); it.hasNext(); ) {
                               //statements here
Example 1:
                    for ( Iterator<String> it = collection.iterator(); it.hasNext();) {
                               while (!condition(it.next())) {
                                         it.remove();
Example 2:
          Iterator<String> it = collection.iterator();
          while (it.hasNext()) {
                     if ( !condition(it.next()) ) {
                               it.remove();
```

Activity Iterator<E> Loop

- Abra o arquivo 'AddWholeNum.java' do package sef.module3.activity
- Faça o seguinte:
 - Usando o List <Integer> anterior, obtenha o objeto iterador dele;
 - Usando um loop for-each, remova todos os valores ímpares;



Summary of For Loops

 Tradicional for é indicado apenas para iterar sobre a matriz. Portanto, nunca faça o seguinte for. Dentro do get () há outra iteração:

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

- O for-each é usado para todas as coleções, incluindo matrizes.
 - Você não pode usá-lo em qualquer lugar. Veja detalhes nos comentários de cada loop.
- O Iterador <E> é indicado para remover elementos da coleção seletivamente.

While Loop

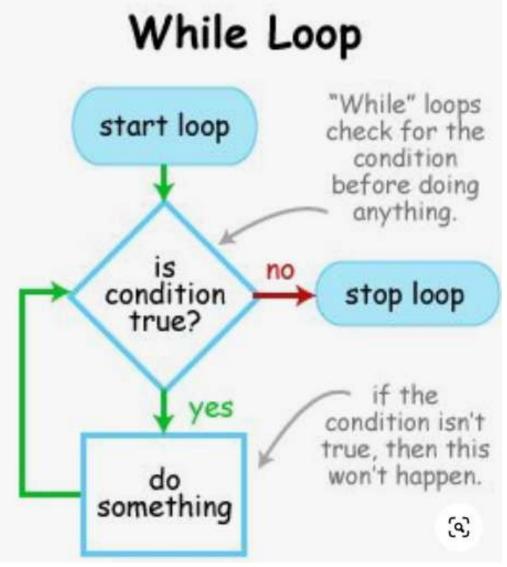
 while() executa instruções repetidamente enquanto uma condição permanece verdadeira

i

Refer to the WhileLoopSample.java sample code.

While Loop

 while() executa instruções repetidamente enquanto uma condição permanece verdadeira



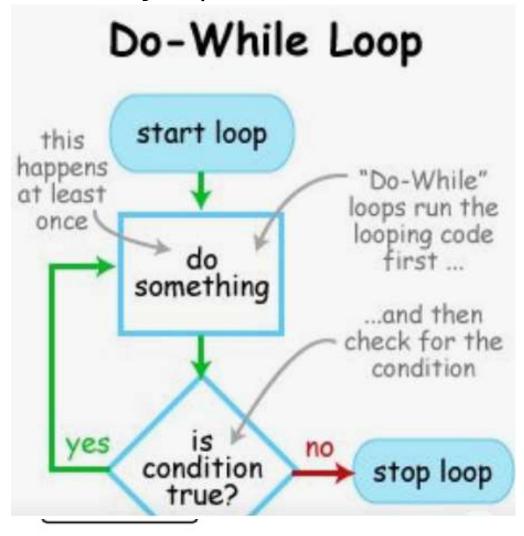
Activity

- 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

While vs do .. while



Do-While Loop

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

```
Syntax:

do {
    //place statements here
} while (condition)

Example:

int x = 1;

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

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

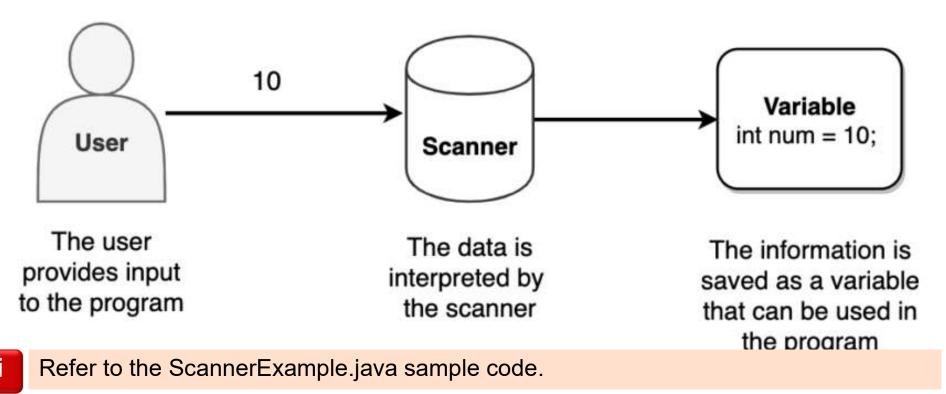
Scanners

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



Scanner

Tipos de input

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.





Scanners

Atividade 2

O programa LocalizaNúmero.JAVA:

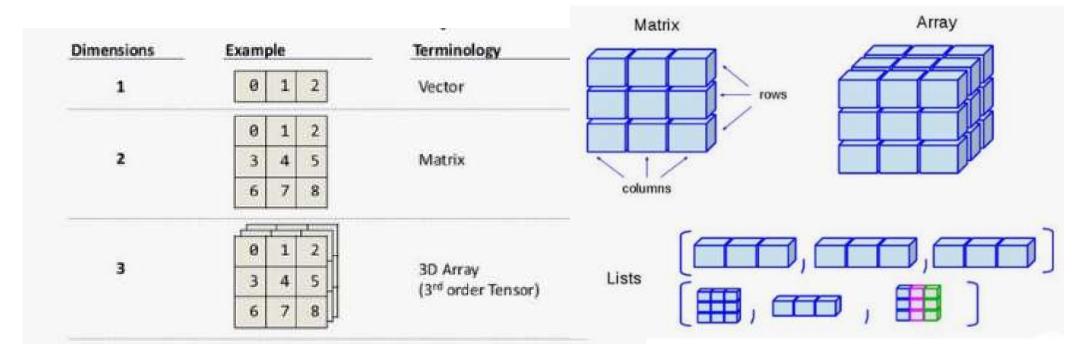
- 1. Corrigir o bug do programa.
- 2. Usar o scanner.

```
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;
          if (achou) {
              System.out.println("Achei");
              System.out.printf("Na Posição %d está localizado do numero %d.", i, vetor[i]);
```

Copyright © 2023 Accer

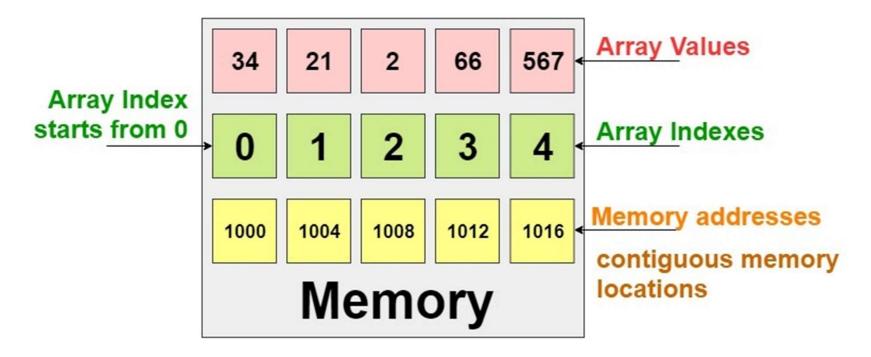
"Arrays"

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

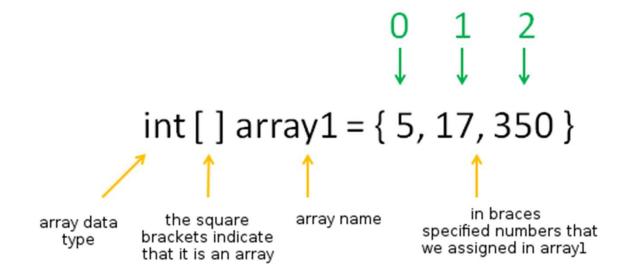


Sintaxe Básica

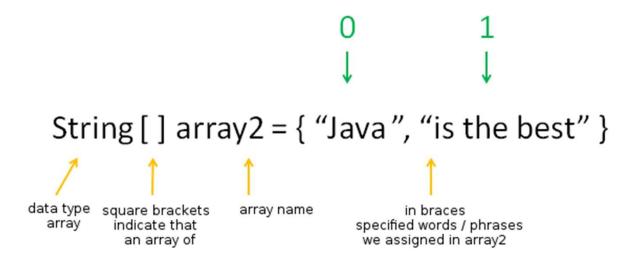
int $x[] = new int[] {34, 21, 2, 66, 567};$



Sintaxe básica:



Sintaxe básica:

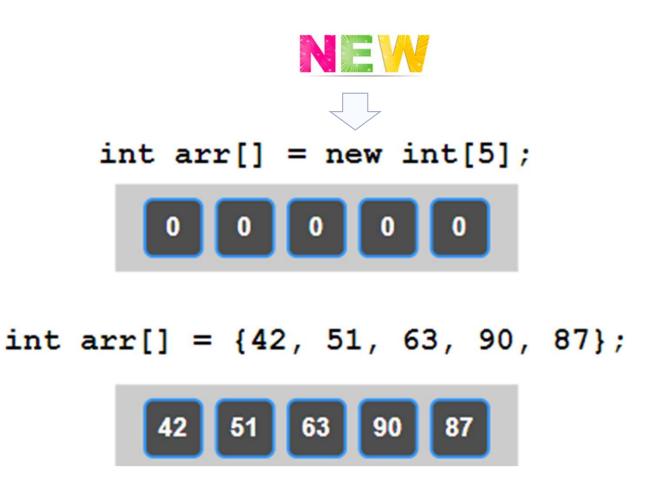


Sintaxe básica:

```
int [] array3 = { 1, 8, 15, 30, 2, 3 };
byte [] array5 = { 1, 3 };
String [] citiesArray = { "Kiev", "Lviv", "Lutsk", "Odessa"};
```

Sintaxe básica:

Como criamos um array vazío?



Sintaxe básica:

Como criamos um array vazío?

```
NEW
   class Test
      public static void
          int[] array1 = new int[3];
          array1[0]=5;
          array1[1]=17;
          array1[2]=350;
10
          System.out.println(array1[0]);
          System.out.println(array1[1]);
13
          System.out.println(array1[2]);
15
```

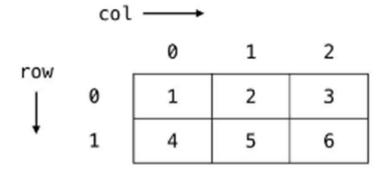
Activity

- 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



Arrays: Multi-Dimensional Arrays

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



Refer to the MultiDimensionalArraySample.java sample code.

Arrays: Manipulation

	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]

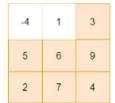
	Column	Column	Column	Column
	1	2	3	4
Row 1	1 a[0][0]	2 a[0][1]	3 a[0][2]	
Row 2	4	5	6	9
	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	7 a[2][0]			

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)



2 at (2,0)

Scanners





Atividade 3

O programa JogodaVelha.JAVA:

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

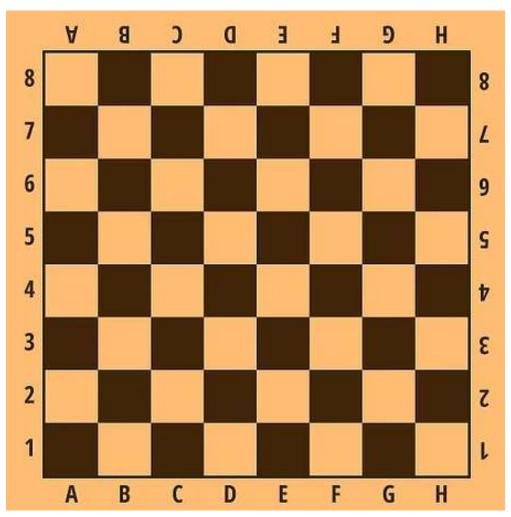


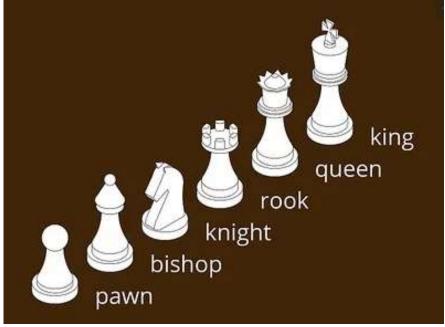
Atividade 4

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

<u>VM1</u>								VM106
(index)	0	1	2	3	4	5	6	7
0	"R"	"N"	"B"	"Q"	"K"	"B"	"N"	"R"
1	"P"	"P"	"P"	"P"	"P"	"P"	"P"	"P"
2	пп	n n			n n		n n	11 11
3	n n							
4	n n	n, n			"p"		0.0	
5	n n							
6	"p"	"p"	"p"	"p"	n n	"p"	"p"	"p"
7	"r"	"n"	"b"	"q"	"k"	"b"	"n"	"r"

i





i

board[x][y] destino <- board[x][y] origem
board[x][y] origem <- apagar;</pre>

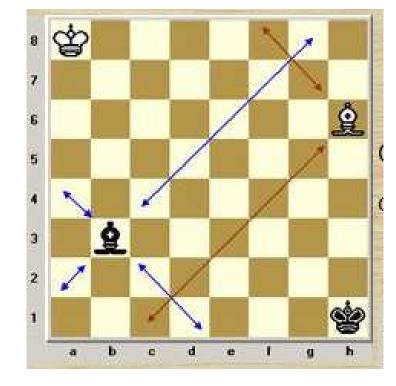


	[0]	[1]	[2]	[3]	[4]	[5]
[0]	[0][0]	[0][1]	[0][2]	[0][3]	[0][4]	[0][5]
[1]	[1][0]	[1][1]	[1][2]	[1][3]	[1][4]	[1][5]
[2]	[2][0]	[2][1]	[2][2]	[2][3]	[2][4]	[2][5]
[3]	[3][0]	[3][1]	[3][2]	[3][3]	[3][4]	[3][5]



 O bispo se movimenta na diagonal mantendo-se sempre nas casas de mesma cor que se encontrava no início do jogo, podendo ir para frente e para trás, quantas casas quiser, mas não pode pular nenhuma outra

peça;

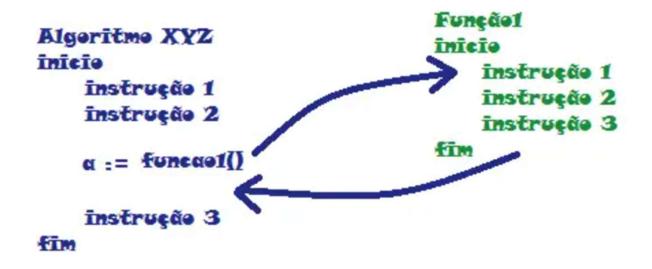


i

"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



Em uma calculadora quais são as suas 4 principais operações?

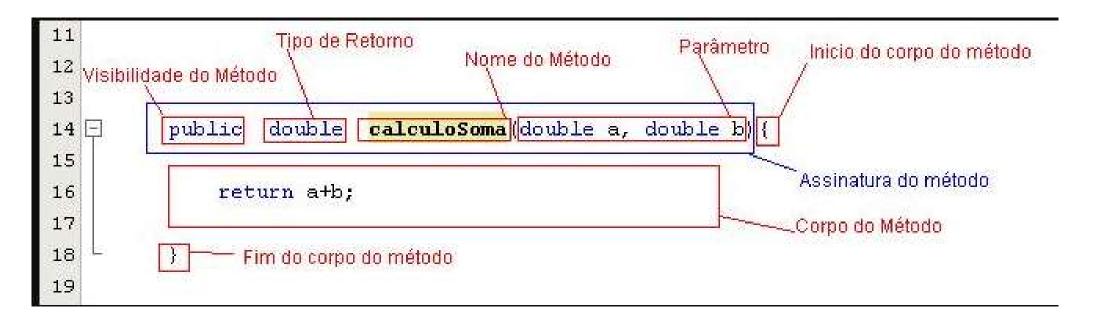


Conseguimos definir as 4 principais operações da calculadora.

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

```
private static int add(int x, int y) {
       public static void main(String[] args)
110
                                                                31
                                                                         int sum = x + y;
 12
                                                                32
                                                                         return sum;
 13
           int x = -11;
                                                                33
           int y = 5;
 14
 15
                                                                     private static int subtract(int x, int y) {
                                                               35∘
           int[] nums = { 1, 2, 3, 4 };
 16
                                                                         int diff = 0;
 17
                                                               37
                                                                         if (x > y) {
           System.out.println("Addition - " + add(x, y));
 18
                                                                            // complete the code
                                                               38
           System.out.println("Subtraction - " + subtract(x, y);
                                                               39
                                                                            diff = x - y;
           System.out.println("Multiply - " + multiply(nums))
 20
                                                               40
                                                                         } else {
          x = 3;
                                                               41
                                                                            // complete the code
 22
          y = 0;
                                                                            diff = y - x;
           System.out.println("Divide - " + divide(x, y));
 23
                                                               43
44
 24
                                                                         return diff;
```

- 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;



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

returns an integer value.

type. This requires the method

to have a 'return' statement that

Calling Methods

- Para "chamar" um método, use o nome do método e passe o número e o tipo de parâmetros apropriados, de acordo com à assinatura do método.
- Ao chamar um método, o controle de fluxo "pulará" para esse método.
- Quando o método terminar, o controle de fluxo retornará ao ponto em que o método foi chamado.
- Os métodos de instância são anexados a uma instância de um objeto. Eles precisarão ser chamados por meio de uma referência de objeto.
- Os métodos estáticos são anexados a uma classe e podem ser chamados pelo nome da classe

Parameter Passing

- Parâmetros em Java são passados por valor
- Passando argumentos de tipo de dados primitivos
 - Uma cópia do valor é passada para o método
 - Quaisquer alterações no valor existem apenas dentro do escopo do método.
 - Quando o método retorna, quaisquer alterações no valor são perdidas. O valor original permanece.
- · Passando argumentos de tipo de dados de referência
 - Uma cópia do valor da referência do objeto está sendo passada.
 - Os valores dos campos do objeto podem ser alterados dentro do método, se eles tiverem o nível de acesso adequado.
 - Quando o método retorna, a referência passada ainda faz referência ao mesmo objeto de antes. No entanto, as alterações nos campos
 κωρομίετο serão mantidas.

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));
```

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

Resumo dos comandos



- Azul Seleção
- Vermelho- Iteração
- Verde Modularização

Activity

- 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;



Activity

- 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



Questions and Comments

 What questions or comments do you have?



