
import org.jetbrains.annotations.NotNull;
import java.util.function.BiConsumer;
/unused, Convert2MethodRef/
public class Java11 {
private void localVariableSyntaxForLambdaParameters() {
// these two are both valid for Java 8 onwards
BiConsumer<Processor, String> consumer = (Processor x, String y) -> x.process(y);
BiConsumer<Processor, String> consumer2 = (x, y) -> x.process(y);
// now valid in Java 11
BiConsumer<Processor, String> consumer3 = (var x, var y) -> x.process(y);
// which is useful for using with annotations
BiConsumer<Processor, String> consumer6 = (@NotNull Processor x, String y) -> x.process(y);
}
Declaration
Console
Description
other
cleanEclipseClasspath
cleanEclipseJdt
cleanEclipseProject
compileJava
compileTestJava
eclipseClasspath
eclipseJdt
eclipseProject

GEEaD - Grupo de Estudos de Educação a Distância

Centro de Educação Tecnológica Paula Souza

São Paulo – SP, 2019

Expediente

PROGRAMA NOVOTEC VIRTUAL

GOVERNO DO ESTADO DE SÃO PAULO

EIXO TECNOLÓGICO DE INFORMAÇÃO E COMUNICAÇÃO

QUALIFICAÇÃO PROFISSIONAL DE ASSISTENTE DE DESENVOLVIMENTO DE SISTEMAS

LÓGICA DE PROGRAMAÇÃO

PÚBLICO ALVO: ALUNOS DA 3ª SÉRIE DO ENSINO MÉDIO

TEMPO DE INTEGRALIZAÇÃO: 34 SEMANAS

Autores:

Eliana Cristina Nogueira Barion

Marcelo Fernando Iguchi

Paulo Henrique Mendes Carvalho

Rute Akie Utida

Revisão Técnica:
Sandra Maria Leandro

Revisão Gramatical:
Juçara Maria Montenegro Simonsen Santos

Editoração e Diagramação:
Flávio Biazim

AGENDA 7

ESTRUTURAS DE REPETIÇÃO II

MERGULHANDO NO TEMA...

Na agenda anterior, você conheceu o comando “para...fim-para”, utilizado em situações em que você sabe exatamente o número vezes que as instruções deverão ser repetidas. Como exemplo, você estudou programas que realizam instruções repetidamente para exibir aos usuários os números de 0 a 9.

Mas, como fazer em situações nas quais não é possível definir quantas vezes a instrução deverá ser repetida?

Os Laços de Repetição abordados nessa agenda são utilizados nesses contextos. São comandos que só encerram suas repetições quando a condição é satisfeita. Dê uma olhada:

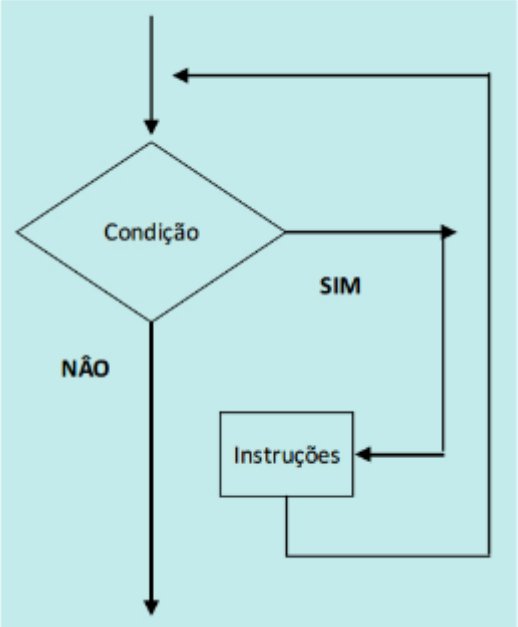


Imagem 01: Representação básica de um fluxograma de Estrutura de Repetição Enquanto (while)

Essa estrutura, também conhecida por **ENQUANTO** (**while**), avalia a condição e só então executa as instruções indicadas. Veja o fluxograma que representa o comando.

Agora o “REPITA... ATÉ QUE” conhecida também como **do...While**. Esta estrutura é muito parecida com a primeira e por conta disso existem vários Programadores que não a usam.

Esta estrutura é utilizada quando, também, não conhecemos o número de repetições que ocorrerá e só encerra, também, quando uma condição é satisfeita. A diferença desta para a primeira é que a condição é testada por último fazendo com que **ao menos uma vez o trecho seja executado**.

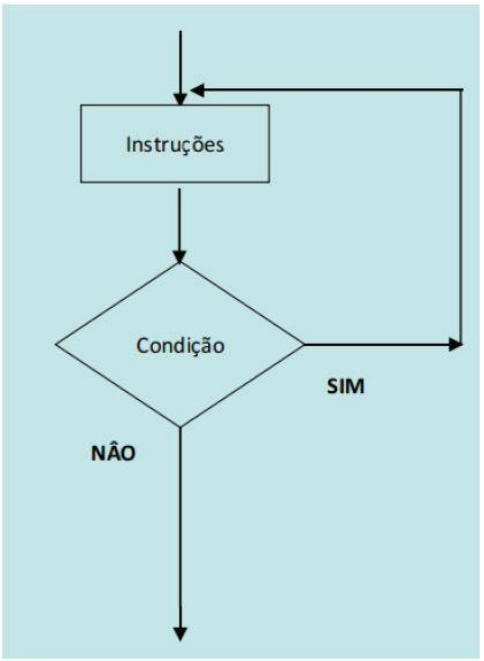
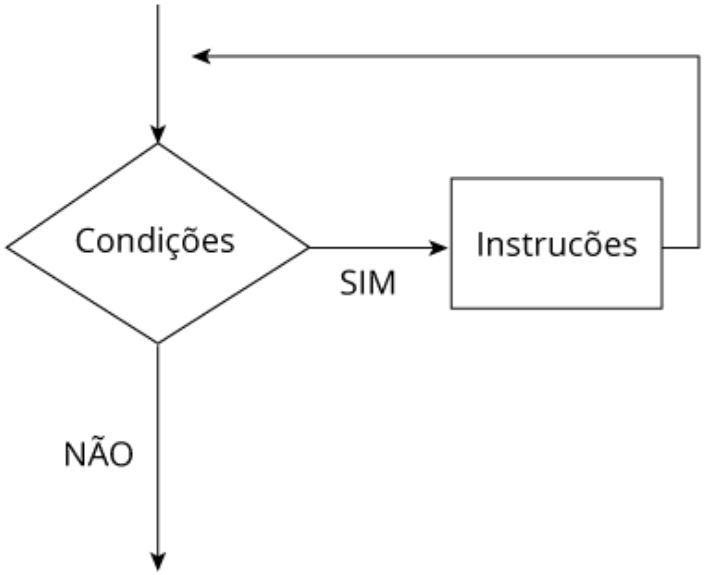


Imagem 02: Representação básica de um fluxograma de Estrutura de Repetição Repita...até que (do while).

Agora, você conhecerá cada Estrutura de Repetição com maior profundidade.

"ENQUANTO... FIM-ENQUANTO"

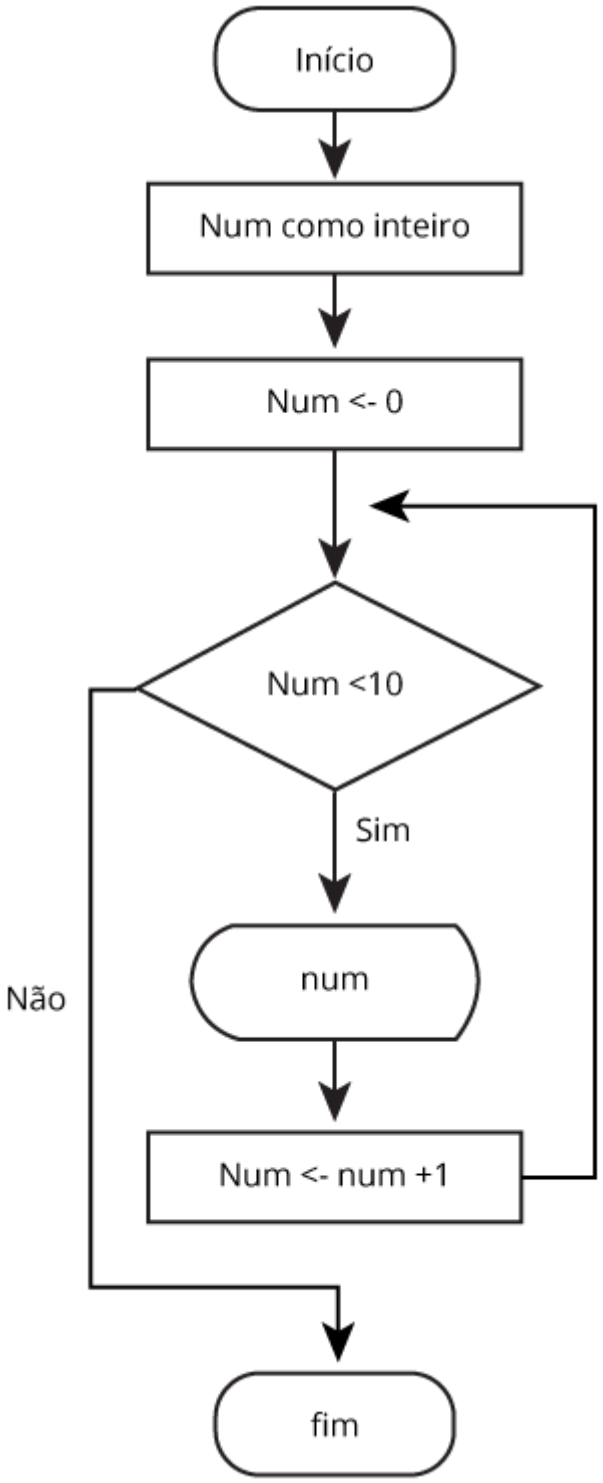
Este Laço de Repetição, como informado anteriormente, trabalha enquanto a condição for verdadeira e vai executando as instruções. Porém, a condição sendo falsa, ele sai do loop e vai para o próximo comando na programação.

PSEUDOCÓDIGO	FLUXOGRAMA	JAVA
Enquanto <condição> **faça** <Comando> <Comando> <Comando> **Fim enquanto**		**while (condição){** instrução **}**

Agora que você conheceu a definição e a estrutura do laço, veja o exemplo a seguir.

Elabore um Algoritmo, um Fluxograma e um Programa em Java que mostre todos os números menores que 10.

Imagem 04: Display de tela

PSEUDOCÓDIGO	FLUXOGRAMA
Programa ex Declare num como inteiro Início Num <- 0 Enquanto (num < 10) faça escreva num num <- num+1 fim-enquanto	

Agora veja em Java:

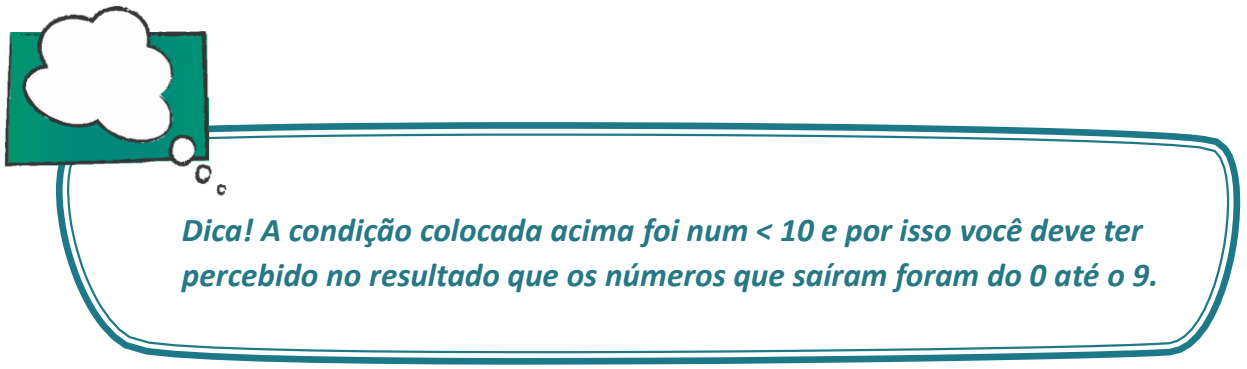
```
1- public class ex01 {
2-
3- public static void main(String[]args) {
4- int num =0;
5- while (num < 10) {
6- System.out.println("Número " + num);
7- Num++;
8-          }
9-      }
```

Observe que na linha 5 temos a estrutura de repetição Enquanto (while) tendo como condição a situação que o exercício colocou (números menores que 10). As linhas 6 e 7 são exatamente os comandos que são executados dentro desta estrutura.

Veja o Resultado apresentado pelo computador:

```
Número 0
Número 1
Número 2
Número 3
Número 4
Número 5
Número 6
Número 7
Número 8
Número 9
```

Imagem 07: Representação do Resultado do Programa em Java que mostra todos os números menores que 10 (Número 0, Número 1, Número 2, Número 3, Número 4, Número 5, Número 6, Número 7, Número 8, Número 9).



O Laço de repetição Enquanto...Fim-Enquanto não possui incremento automático como o laço Para...Fim...Para.

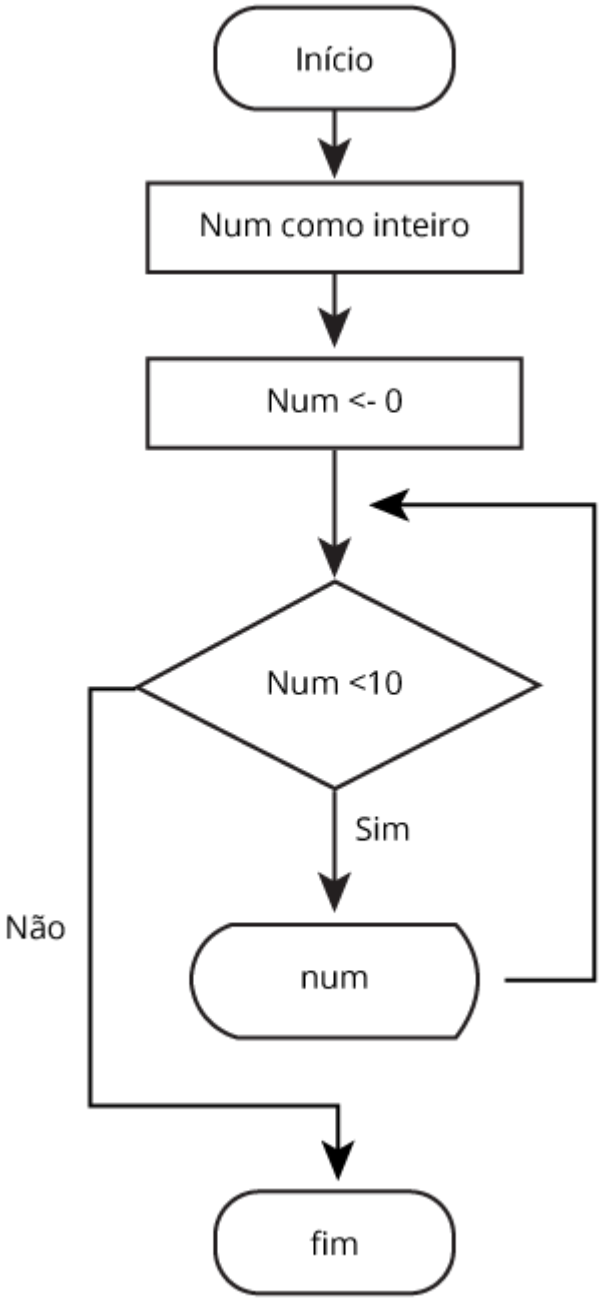
Fique atento a isso!

O exemplo a seguir apresenta uma repetição infinita e demonstra as implicações da ausência de contador. Veja:

PSEUDOCÓDIGO	FLUXOGRAMA

```
Programa ex

   Declare
        num como inteiro
   Início
        Num <- 0
            Enquanto (num < 10) faça
             escreva num
        fim-enquanto
```



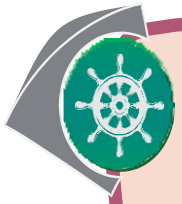
Veja como fica em Java:

```
1- public class exemploLoopInfinito {
2-
3- public static void main(String[]args) {
4- int num =0;
5- while (num < 10) {
6- System.out.println("Número " + num);
7-          }
8-       }
9- }
```

Note que o programa é idêntico ao anterior com a exceção de que não temos o incremento do contador num (num = num+1). Com isso, o contador fica eternamente com o valor de 0 (zero) e o programa nunca terá um fim.

Observe o Resultado:

```
Número 0
Número 0
Número 0
Número 0
Número 0
Número 0
Número 0
Número 0
Número 0
Número 0
......
Infinita vezes
```



VOCÊ NO COMANDO

Você viu que o comando ENQUANTO é muito simples. Como no exemplo acima, foram mostrados apenas os números de 0 até 9, será que você consegue fazer com que mostre no Java os números de 0 até 10?

Dica: este assunto já foi abordado na agenda anterior!

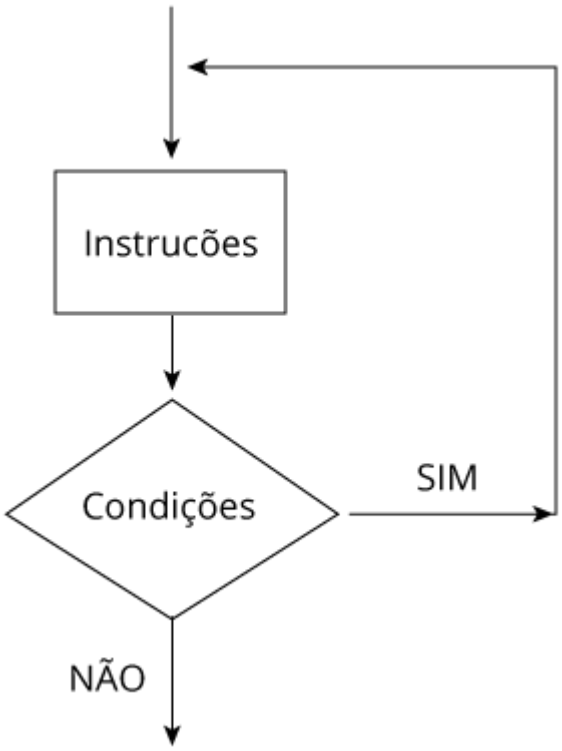
Confira com a codificação a seguir:

```
1- public class ex01 {
2-
3- public static void main(String[]args) {
4- int num =0;
5- while (num < = 10) {
6- System.out.println("Número " + num);
7- Num++;
8-          }
9-        }
10-     }
```

A diferença deste código no Java com o anterior é apenas o operador igual (=). Ao colocar a condição <= 10 você incluiu o 10 na condição e o resultado (mostrando a seguir) aparecem os números de 0 a 10. Resultado:

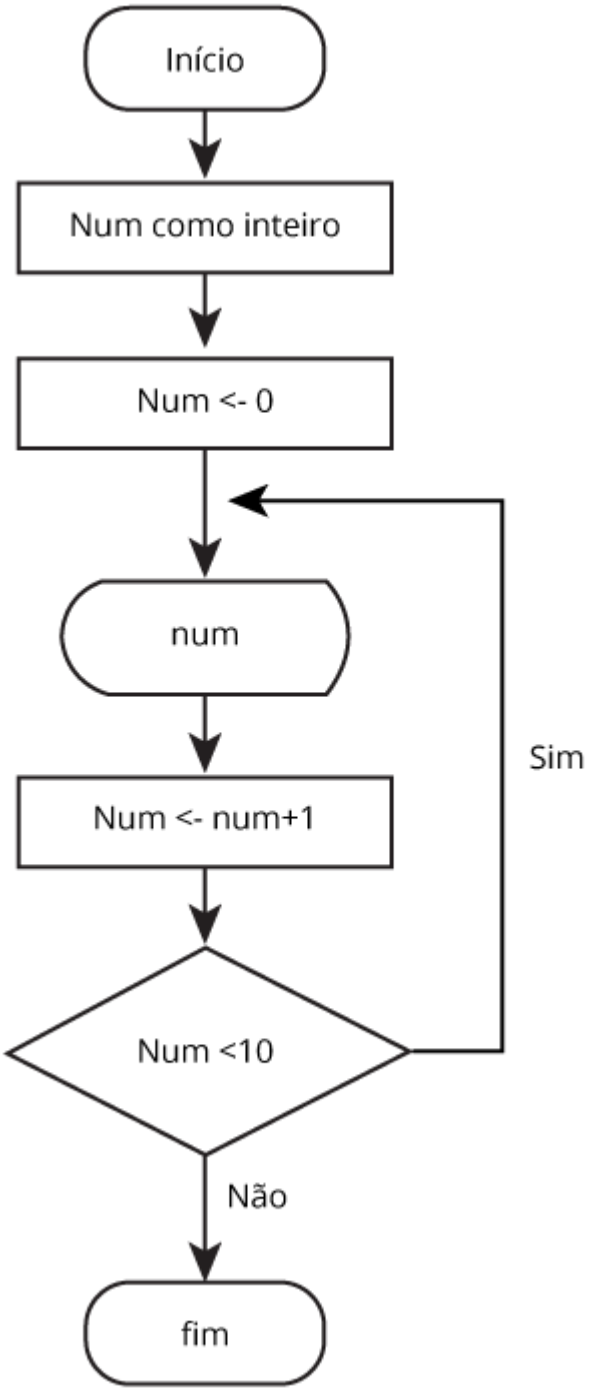
```
Número 0
Número 1
Número 2
Número 3
Número 4
Número 5
Número 6
Número 7
Número 8
Número 9
Número 10
```

"REPITA... ATÉ QUE"

PSEUDOCÓDIGO	FLUXOGRAMA	JAVA
Repita <condição> **faça** <Comando> <Comando> <Comando> **até que**<Comando>		**do {** instrução **}** **while** (condição)

Como você já sabe, esta estrutura é parecida com a while, que você acabou de conhecer. A única diferença é que a condição dela é executada por último. Vamos aos detalhes para entender melhor?

Agora que você conheceu um pouco mais esta estrutura, vamos voltar ao exemplo anterior.

PSEUDOCÓDIGO	FLUXOGRAMA
Programa ex01 Declare num como inteiro Início repita escreva num num < num + 1 até que (num < 10) fim.	

Observe como fica em Java:

```
1-
2- public class ex01 {
3-
4- public static void main(String[]args) {
5- int num =0;
6- do{
7- System.out.println("Número " + num);
8- Num++;
9-  }
10-     while (num < 10);
11-            }
12-     }
```

Observe que, como o programa responde ao mesmo exercício, e o resultado é o mesmo. Porém, o programa foi construído utilizando outro Laço de repetição. Note que as linhas 6 até 10 apresentam a sintaxe do comando **repita até.**

Veja na videoaula a seguir, que o professor Sandro Valérios retoma os conteúdos estudados até o momento.

Utilizando comandos de repetição em conjunto com comando condicional if

Nas agendas anteriores você viu que é possível utilizar o comando para (for) em conjunto com o se (if) para atingir a solução de um problema na programação. Assim, como no caso do para, também podemos usar o enquanto (while) e o repita até (do while) em conjunto com o se (if) com o mesmo propósito.

Veja o exemplo a seguir, um programa de uma empresa financeira. Nele, o usuário informa seu cargo e salário e, a partir dessas informações, o programa informa quanto de empréstimo a pessoa pode obter.

diretor = 30%	Gerente = 25%	Operacional = 20%

```
1- Import java.util.Scanner;
2- public class ex03 {
3-
4- public static void main(String[]args) {
5-
6- String nome, cargo;
7- float sal, emprestimo=0;
8-
9-  System.out.println("Por favor informe seu nome");
10-      nome = new Scanner(System.in).nextLine();
11-      System.out.println("E agora informe seu cargo (Diretor, Gerente
   ou Operacional);
12-      cargo = new Scanner(System.in).nextLine();
13-
14-      while (cargo == "Diretor" || cargo == "Gerente" || cargo ==
   "Operacional" {
15-      System.out.println("Cargo incorreto, por favor informe o cargo
   novamente");
16-      cargo = new scanner(System.in).nextLine();
17-      }
18-
19-      System.out.println("Agora informe o seu salário")
20-      sal = new scanner(System.in).nextFloat();
21-      if (cargo.equals("Gerente"))
22-      emprestimo = sal*25/100;
23-      else if (cargo.equals("Diretor"))
24-      emprestimo = sal*30/100;
25-       else emprestimo = sal*20/100;
26-      System.out.println("Olá" + nome);
27-      System.out.println("Seu cargo é" + cargo);
28-      System.out.println("Seu salário é" + sal);
29-      System.out.println("Olá" + nome);
30-      System.out.println("E você tem direito a pegar R$" + emprestimo
   + "de mepréstimo");
31-      }
32-      }
```

No exemplo acima, observamos que além dos usos do Laço de repetição WHILE (linhas 14 até 17), da estrutura de seleção IF (linhas 21 até 26), foi utilizado, também, o scanner (linhas 1, 10, 12, 16 e 21). Pois é, neste momento do curso você já tem diversos comandos em seu repertório, e deve utilizá-los conjuntamente para solucionar os desafios propostos.

VOCÊ NO COMANDO

1. Faça o Fluxograma e um Programa em Java que leia 50 números, calcule e exiba a média aritmética dele.

2. Faça um Programa em Java para calcular a soma dos dígitos de um número.

Por exemplo: N°: 21 -> 2+1 = 3

3. Faça um Programa em Java para realizar comparação de números e informar o número maior. Para este código será necessário solicitar ao usuário digitar a quantidade de números que ele quer comparar e ele deverá digitar os números.

Confira seu fluxograma e programas com as respostas a seguir:

1. Fluxograma:

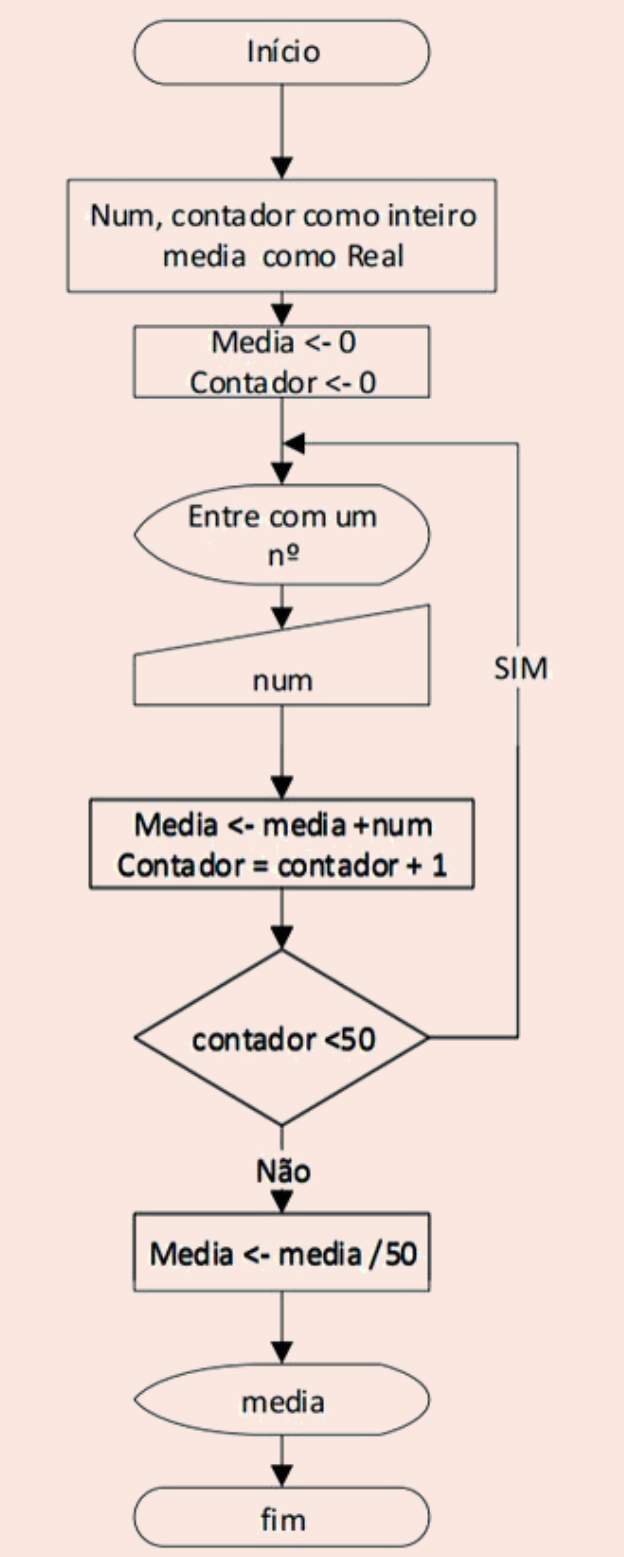


Imagem 12: Resposta do fluxograma do exercício 1:

Veja em Java:

```
1- import javax.swing.JOptionPane;
2-
3- public class Repeticao {
4-
5- public static void main(String[]args) {
6- // exercício 1
7-
8- //declaração de variáveis
9-      int num, contador = 0;
10-     double media = 0;
11-
12-     media = 0;
13-     //repetição
14-     do {
15-     //entrada de dados
16-     Num = Integer.ParseInt(JOptionPane.showInputDialog(Contador +
   "Entre com um número"));
17-     //soma
18-     Media = media + num;
19-     contador++;
20-     }while (num < 50); //fim da repetição
21-     Media = media/50; //cálculo da media
22-     //saída de dados
23-     JOptionPane.showMessageDialog(null, "A media é: " + media);
24-
25-             }// fim do main
26-     }// fim da classe
```

2.

```
1- import javax.swing.JOptionPane;
2-
3- public class somaDigitos {
4-
5- public static void main(String[]args) {
6- //declaração de variáveis
7-      int num, soma = 0;
8- //entrada de dados
9- Num = Integer.ParseInt(JOptionPane.showInputDialog("Entre com um
   número inteiro "));
10-
11-     //salvo o valor das unidades
12-     Soma = (num % 10);
13-
14-     //cálculo a soma das dezenas, centenas,etc …
15-     while (num > 0){
16-     num / = 10;
17-     soma = soma + (num%10);
18-     } //fim da repetição
19-     //saída de dados
20-     JOptionPane.showMessageDialog(null, soma);
21-
22-             }
23-     }
```

3.

```
1- Import java.util.Scanner;
2- public class ex06 {
3-
4- public static void main(String[]args) {
5-
6- int x = 2;//iniciando a variável com valor
7- int num;
8- int maior=0;
9- int compara;
10-
11-     Scanner entrada = new Scanner(System.in);//utilizando scanner
   para capturer dados
12-     System.out.println("Digite quantos números você quer comparar:
   ");
13-     compara = entrada.nextInt();
14-
15-     System.out.println("Insira o primeiro número ", maior);
16-     maior = entrada.nextInt();
17-     while (x <= compara ){ //laço de repetição while
18-     System.out.println("Digite" + x + " número ");
19-     num = entrada.nextInt();
20-     se (num > maior){ //decisão
21-     maior = num;
22-     }
23-     x++;
24-     }
25-     System.out.println("O maior número digitado foi" + maior);
26-     }
27-     }
```