

**targettrust**  
treinamento e tecnologia



# Riguel Figueiró

---

Consultor de software na ThoughtWorks  
Apaixonado por metodologias ágeis,  
integração contínua, qualidade e disseminação  
e compartilhamento de conhecimento!

Como falar comigo:

- [riguel@dkosoftware.com.br](mailto:riguel@dkosoftware.com.br)
- [www.linkedin.com/in/riguel-figueiro/](https://www.linkedin.com/in/riguel-figueiro/)
- [github.com/riguelbf](https://github.com/riguelbf)





# **JAVA DEVELOPER**

## Imersão Completa



# Variáveis primitivas e controle de fluxo



# Tipos primitivos

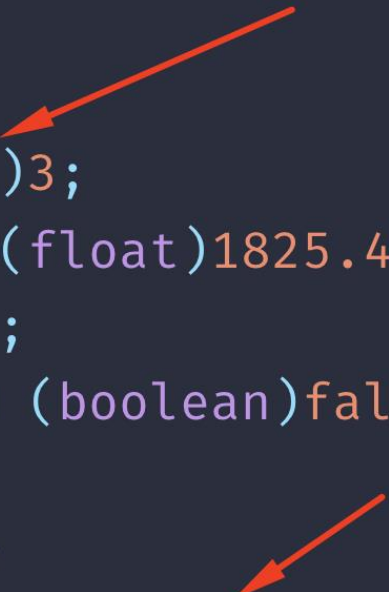
---

- **Inteiros**: Representa valores numéricos negativo ou positivo sem casa decimal, ou seja, valores inteiros.
- **Real**: Representa valores numéricos negativo ou positivo com casa decimal, ou seja, valores reais.

Também são chamados de ponto flutuante

- **Lógico**: Representa valores booleanos, assumindo apenas dois estados, VERDADEIRO ou FALSO. Pode ser representado apenas um bit (que aceita apenas 1 ou 0).
- **Texto**: Representa uma sequência de um ou mais de caracteres, colocamos os valores do tipo TEXTO entre " " (aspas duplas).

```
1  int idade = 3;
2  float salario = 1825.45;
3  char letra = 'G';
4  boolean casado = false;
5
6  // Type casting
7  int idade = (int)3;
8  float salario = (float)1825.45;
9  char letra = 'G';
10 boolean casado = (boolean>false;
11
12 // Wrapper class
13 Integer idade = new Integer(3);
14 Float salario = new Float(1825.45);
15 Character letra = new Character('G');
```



# Tamanho de tipos em bytes

Família	Tipo Primitivo	Classe Invólucro	Tamanho	Exemplo
Lógico	boolean	Boolean	1 bit	true
Literais	char	Character	1 byte	'A'
	-	String	1 byte/cada	"JAVA"
Inteiros	byte	Byte	1 byte	127
	short	Short	2 bytes	32 767
	int	Integer	4 bytes	2 147 483
	long	Long	8 bytes	$2^{63}$
Reais	float	Float	4 bytes	$3.4e^{+38}$
	double	Double	8 bytes	$1.8e^{+308}$

# Controle de fluxo

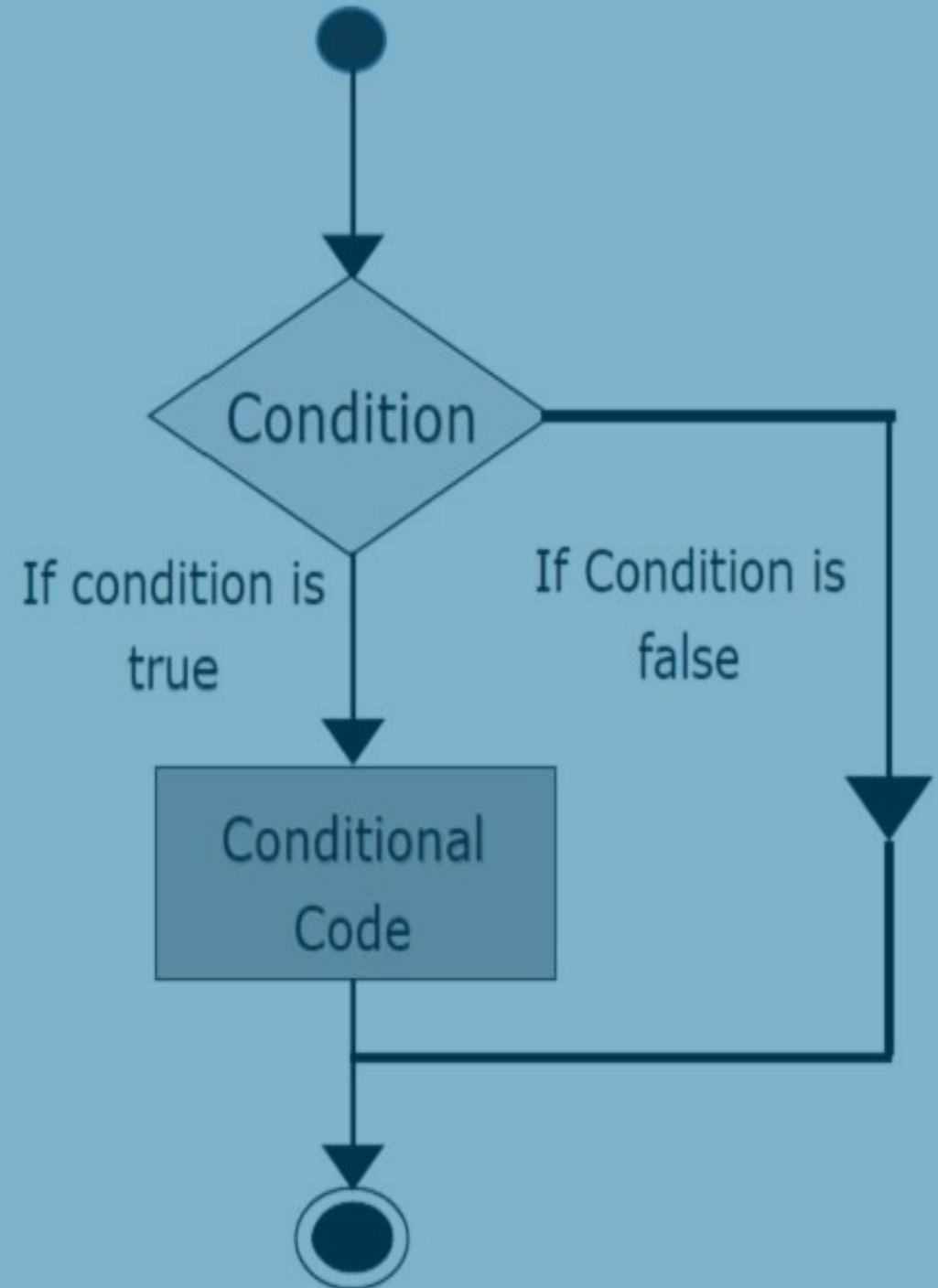
---

... é a habilidade de ajustar a maneira como um programa realiza suas tarefas. Por meio de instruções especiais, chamadas comandos, essas tarefas podem ser executadas seletivamente, repetidamente ou excepcionalmente. Não fosse o controle de fluxo, um programa poderia executar apenas uma única sequência de tarefas, perdendo completamente uma das características mais interessantes da programação: a dinâmica.



# IF - Básico

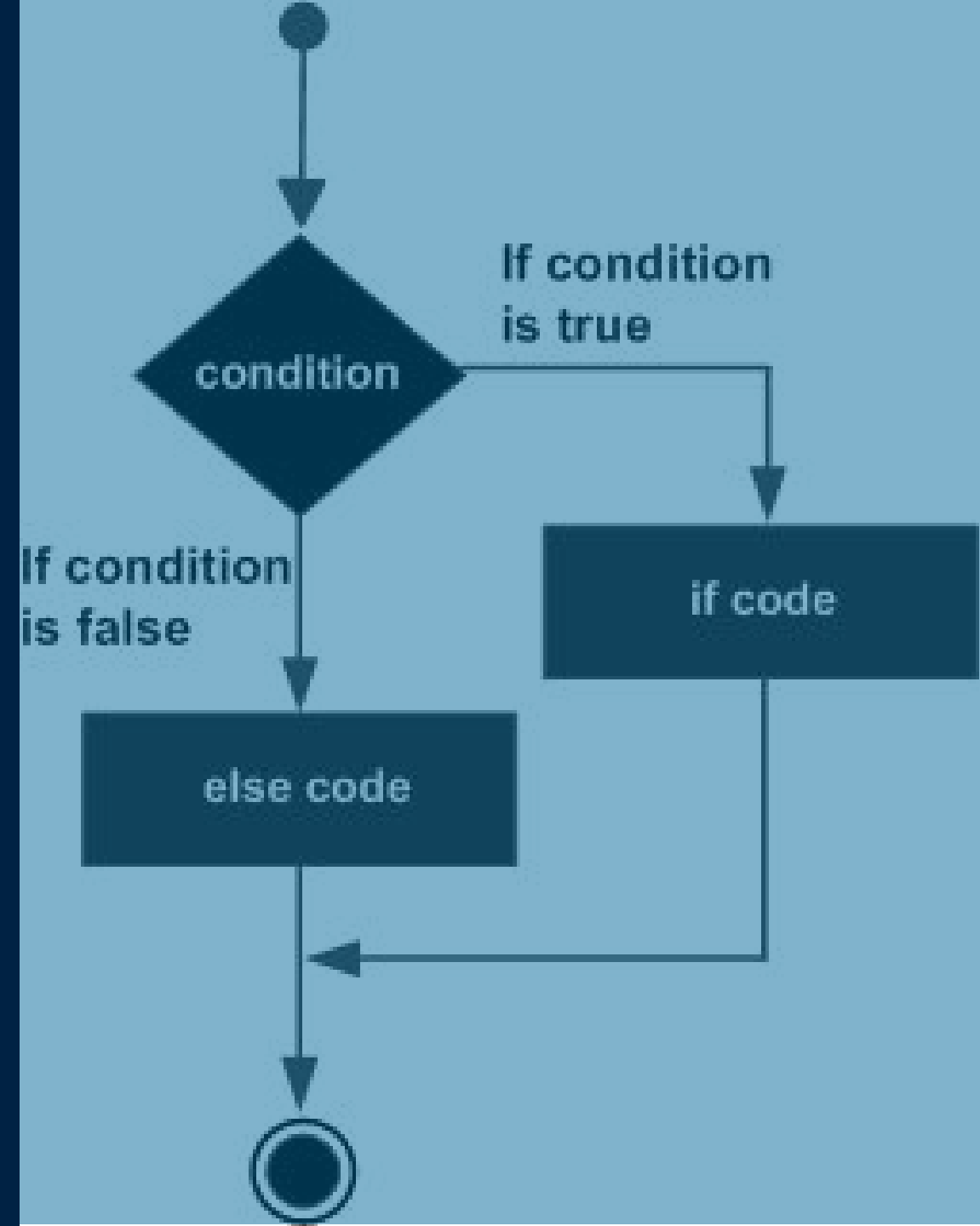
O comando IF é comum em muitas linguagens de programação, sua função é verificar se uma condição é verdade ou falsa





# IF \* ELSE - Básico

Caso a condição verificada dentro da condicional “IF” podemos realizar o encadeamento dentro do “FALSE”



---

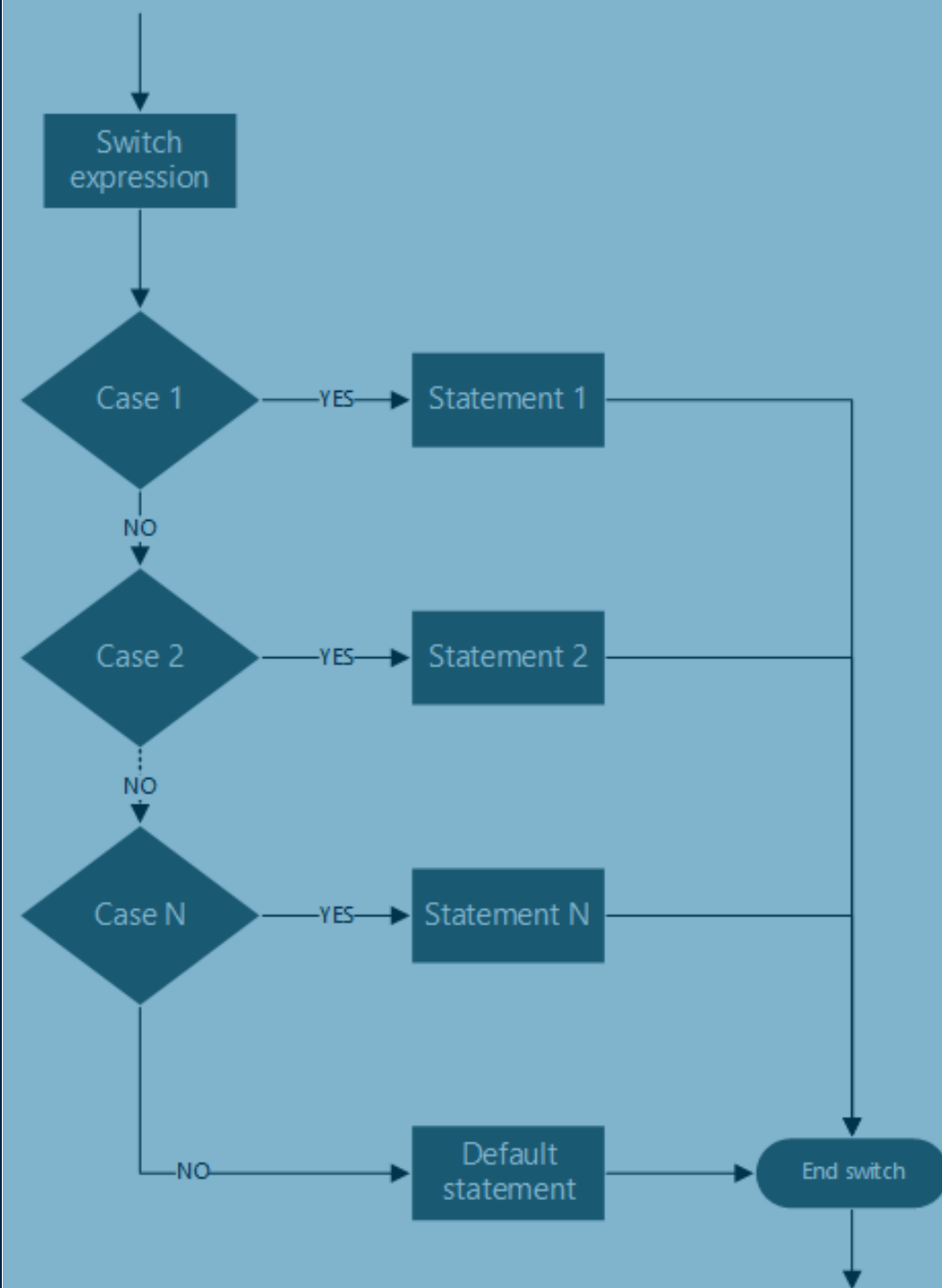
```
1  // A sintaxe do if * else no Java é a seguinte:
2  if (condicaoBooleana) {
3      |    codigo;
4  }else{
5      |    // codigo para condição falsa
6  }
```

```
1  int i = 10;
2      if (i > 10) {
3          if (i == 10) {
4              System.out.println("x igual a 10");
5          }
6      else if (i == i++) {
7          System.out.println("i menor que 10");
8      }
9      else {
10         System.out.println("i não é menor que 10");
11     }
12
```

Exemplo prático

# Switch case

Assim como no caso execução seletiva de múltiplos comandos, há situações em que se sabe de antemão que as condições assumem o valor true de forma mutuamente exclusiva, isto é, apenas uma entre as condições sendo testadas assume o valor true num mesmo momento



```
1  int op = 2;
2
3  System.out.print("valor de op eh: "+op)
4
5  switch(op) {
6      case 1:
7          System.out.println("case 1: op="+op);
8          break;
9      case 2:
10         System.out.println("case 2: op="+op);
11         break;
12     case 3:
13         System.out.println("case 3"+op);
14         break;
15     default:
16         System.out.println("default: op nao esta no limite 1..3");
17 }
```

Exemplo prático



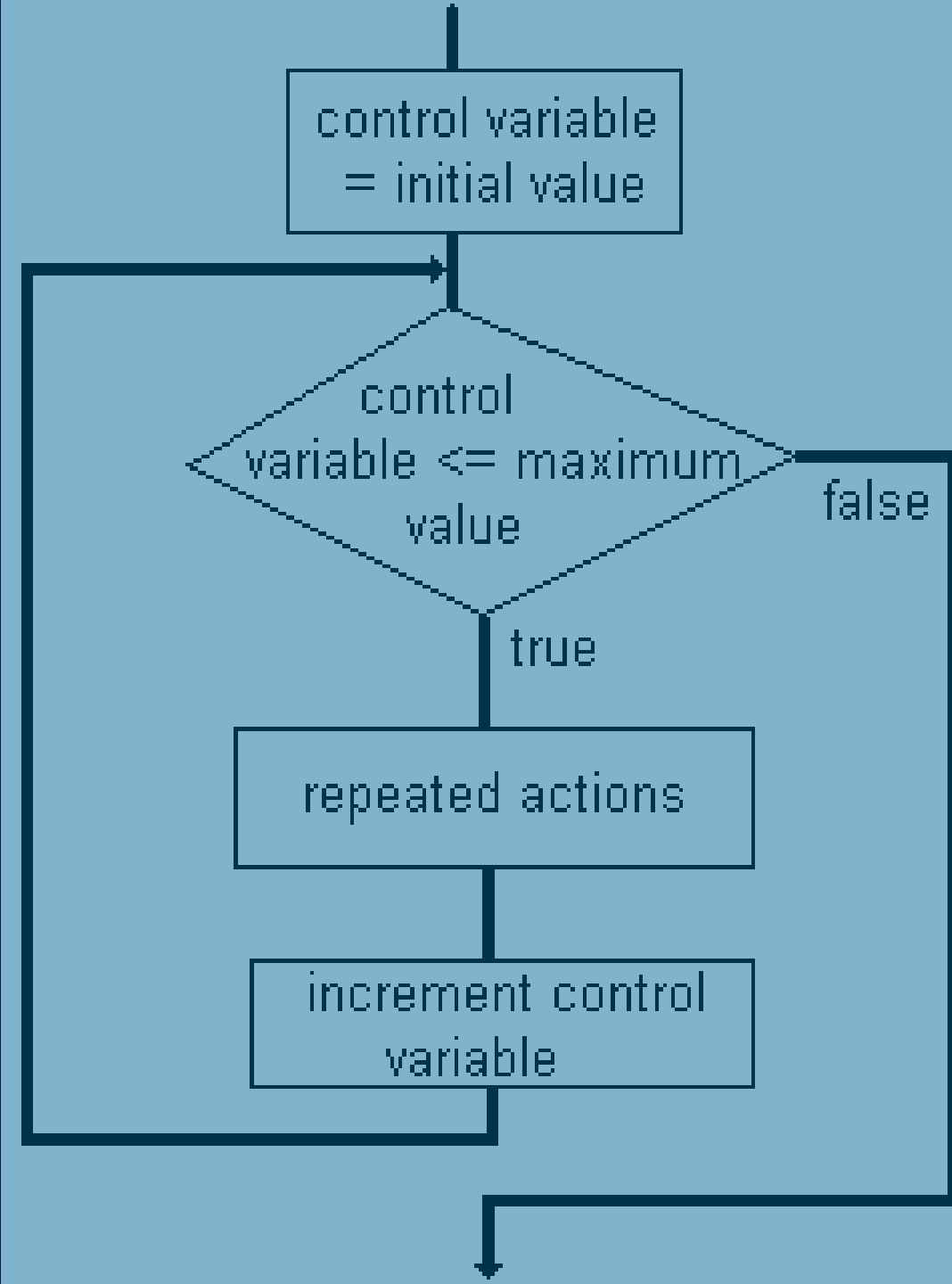
# Loops – estruturas de repetição





# While – estrutura de repetição (enquanto/faça)

... executa uma comparação com a variável. Se a comparação for verdadeira, ele executa o bloco de instruções ( { } ) ou apenas a próxima linha de código logo abaixo.

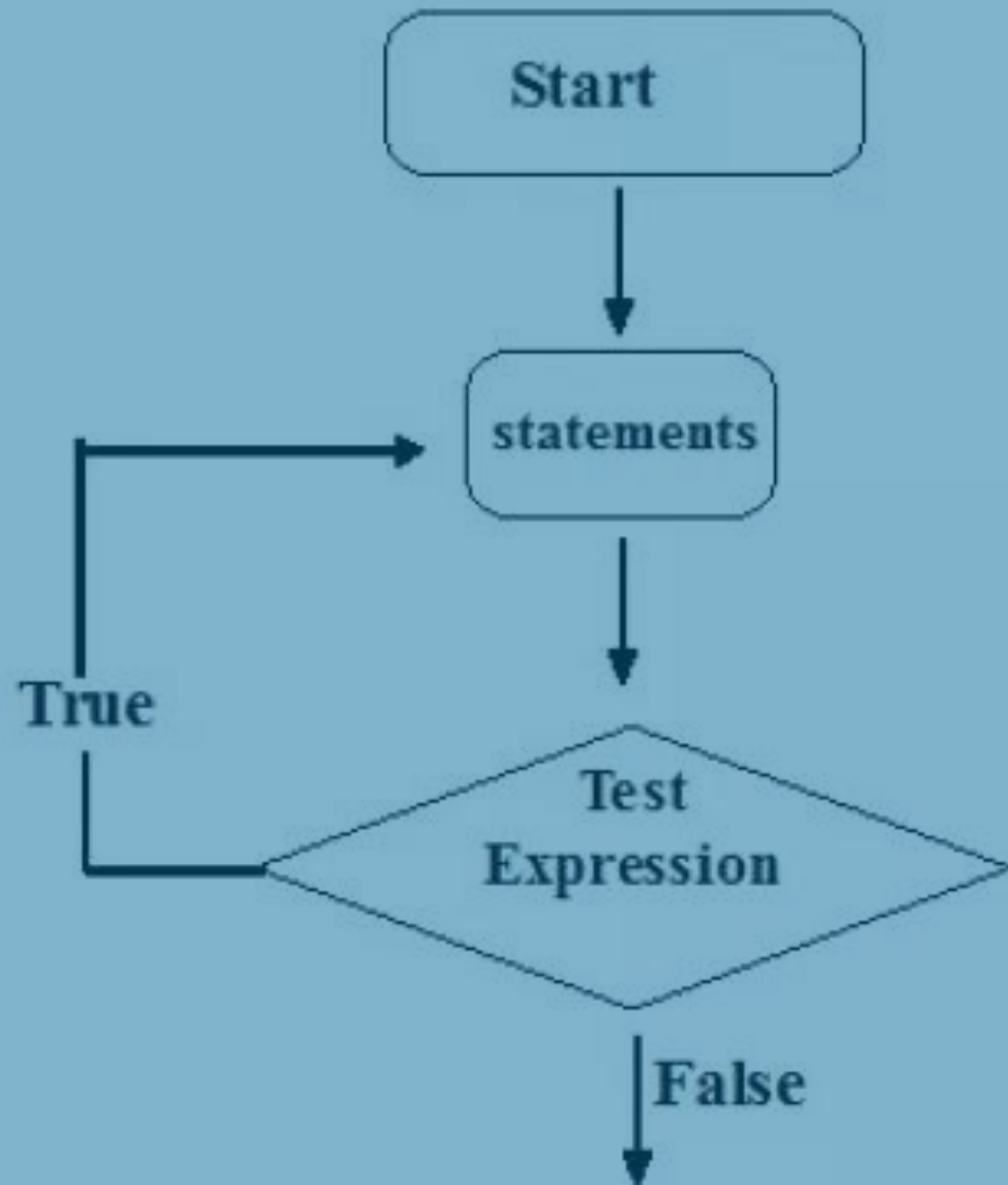


```
double a, b, x=1.5, erro = 0.05;
a = 1; b = 2; // 1 < (raiz de 2) < 2
while( (b-a) > erro ) {
    x = (a+b)/2;
    if (x*x < 2) // x < raiz de 2
        a = x;
    else // x ≥ raiz de 2
        b = x;
}
```

Exemplo prático

# Do While – estrutura de repetição (faça/enquanto)

... é uma estrutura de repetição, tal como o próprio while. A principal diferença entre os dois é que DO WHILE irá fazer a comparação apenas no final do bloco de código, sendo representado da seguinte forma:

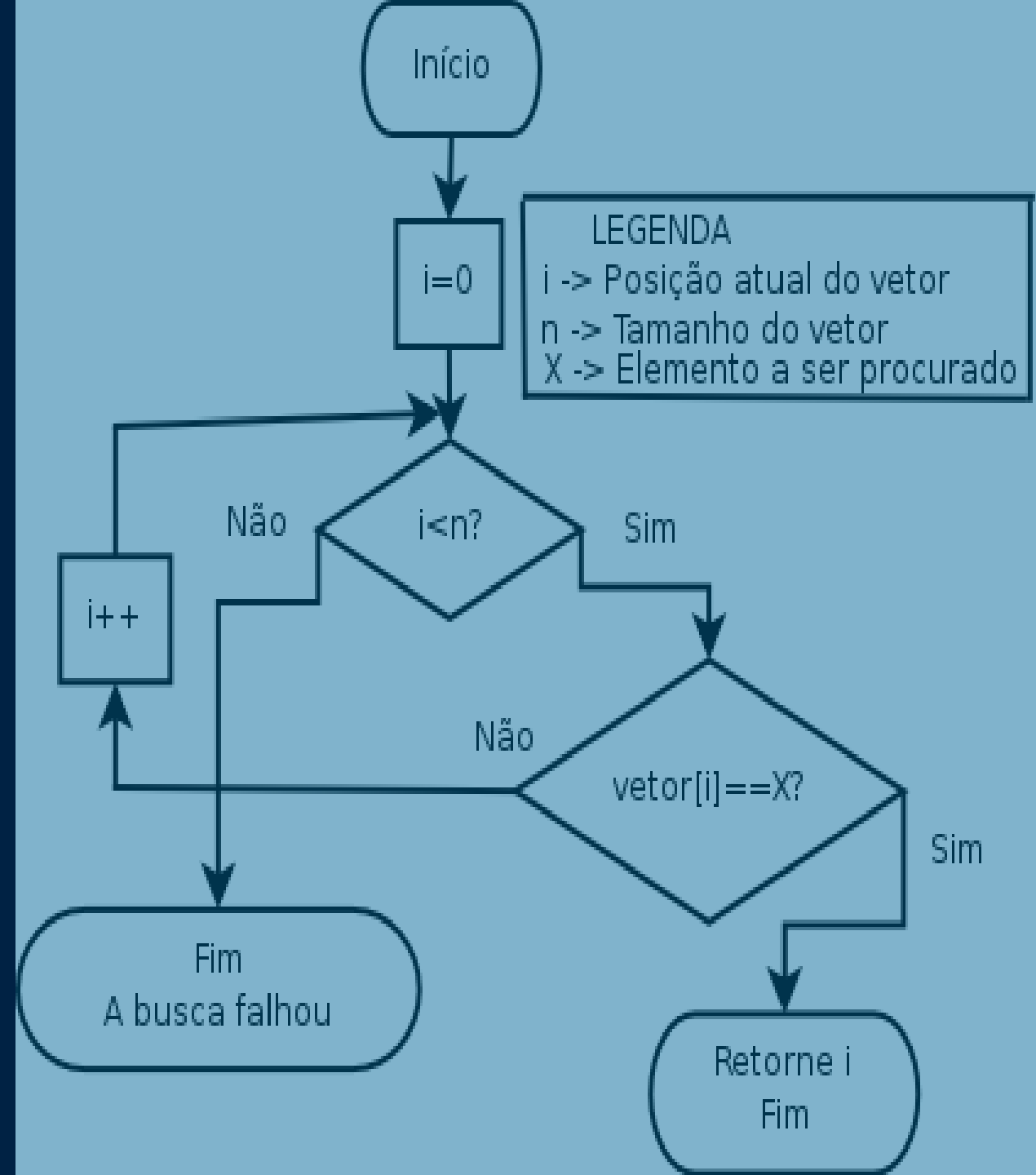


```
1  int counter = 5;
2  int factorial = 1;
3  do {
4      factorial *= counter--; /* Multiply, then decrement. */
5  } while (counter > 0);
6
7  System.out.println("The factorial of 5 is " + factorial);
8
```

Exemplo prático

# For – estrutura de repetição (iteração com contagem)

... for isola também um espaço para inicialização de variáveis e o modificador dessas variáveis. Isso faz com que fiquem mais legíveis, as variáveis que são relacionadas ao loop:



---

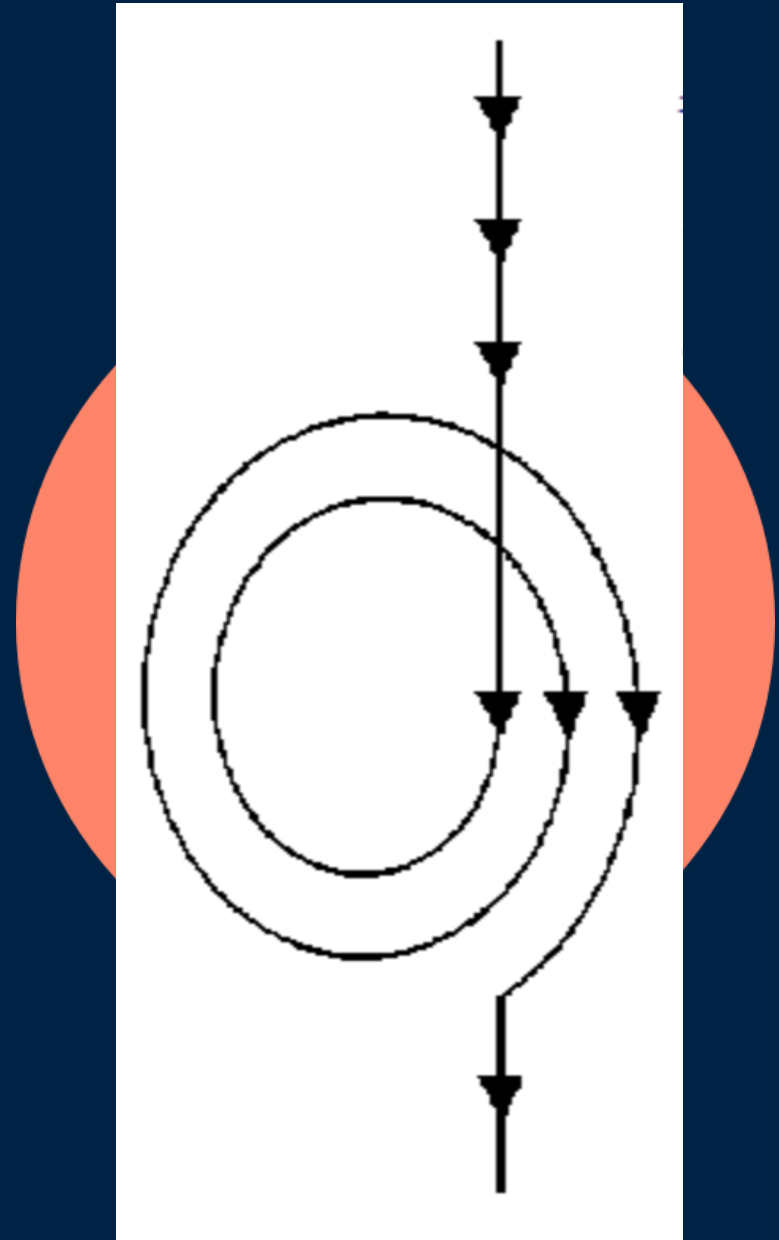
```
1  for (int i = 0; i < 10; i++)    {  
2      System.out.println("olá!");  
3  }
```

Exemplo prático

# CONTROLANDO LOOPS

---

Apesar de termos condições booleanas nos nossos laços, em algum momento, podemos decidir parar o loop por algum motivo especial sem que o resto do laço seja executado.



# Break

---

```
1  for (int i = x; i < y; i++) {
2      if (i % 19 == 0) {
3          System.out.println("Achei um número divisível por 19 entre x e y");
4          break;
5      }
6  }
7  class Main {
8      public static void main(String[] args) {
9          int x = 0;
10         int y = 30;
11         for (int i = x; i < y; i++) {
12             if (i % 19 == 0) {
13                 System.out.println("Achei um número divisível por 19 entre x e y");
14                 break;
15             }
16         }
17     }
18 }
```



# Continue

---

```
1  for (int i = 0; i < 100; i++) {
2      if (i > 50 && i < 60) {
3          continue;
4      }
5      System.out.println(i);
6  }
7
8  class Main {
9      public static void main(String[] args) {
10         for (int i = 0; i < 100; i++) {
11             if (i > 50 && i < 60) {
12                 continue;
13             }
14             System.out.println(i);
15         }
16     }
17 }
```



# Pergunta

---

Façam o máximo de perguntas que desejarem!!



# Exercícios

---

<https://docs.google.com/document/d/1NgomyK92ilyErtdPwVIFUQybkVWGRyQKsFBpIbwRqHs/edit?usp=sharing>

