Introdução ao Java – Parte II

Programação Orientada a Objetos — QXD0007

Prof. Atílio Gomes Luiz
gomes.atilio@ufc.br

Universidade Federal do Ceará
Campus Quixadá

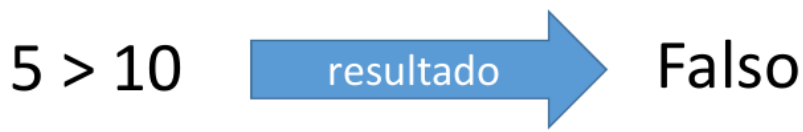
1º semestre/2022

Tópicos desta Aula

- Operadores relacionais
- Operadores lógicos
- Estruturas condicionais
- Operadores de atribuição cumulativa
- Estrutura de repetição
- Funções
- Arrays

Operadores relacionais

Expressões comparativas



Operadores relacionais (comparativos)

Operador	Significado
$>$	maior
$<$	menor
$>=$	maior ou igual
$<=$	menor ou igual
$==$	igual
$!=$	diferente

Operadores relacionais (comparativos)

Operador	Significado
$>$	maior
$<$	menor
$>=$	maior ou igual
$<=$	menor ou igual
$==$	igual
$!=$	diferente

Exemplo: Suponha x igual a 5

- $x > 0$ Resultado: V
- $x == 3$ Resultado: F
- $x <= 30$ Resultado: V
- $x != 2$ Resultado: V

Operadores lógicos

Expressões lógicas e operadores lógicos

Expressões lógicas e operadores lógicos

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Operadores Lógicos:

Operador	Significado
&&	E
||	OU
!	NEGAÇÃO

Significado dos operadores lógicos

Ideia por trás do operador &&

- Você pode obter uma habilitação de motorista se: for aprovado no exame psicotécnico **E** for aprovado no exame de legislação **E** for aprovado no exame de direção.

Significado dos operadores lógicos

Ideia por trás do operador &&

- Você pode obter uma habilitação de motorista se: for aprovado no exame psicotécnico **E** for aprovado no exame de legislação **E** for aprovado no exame de direção.
  - **Todas as condições devem ser verdadeiras**

Significado dos operadores lógicos

Ideia por trás do operador &&

- Você pode obter uma habilitação de motorista se: for aprovado no exame psicotécnico **E** for aprovado no exame de legislação **E** for aprovado no exame de direção.
  - **Todas as condições devem ser verdadeiras**

A	B	A && B
F	F	F
F	V	F
V	F	F
V	V	V

Ideia por trás do operador ||

- Você pode estacionar na vaga especial se: for idoso(a) **OU** for uma pessoa com deficiência **OU** for uma gestante.

Significado dos operadores lógicos

Ideia por trás do operador &&

- Você pode obter uma habilitação de motorista se: for aprovado no exame psicotécnico **E** for aprovado no exame de legislação **E** for aprovado no exame de direção.
  - **Todas as condições devem ser verdadeiras**

A	B	A && B
F	F	F
F	V	F
V	F	F
V	V	V

Ideia por trás do operador ||

- Você pode estacionar na vaga especial se: for idoso(a) **OU** for uma pessoa com deficiência **OU** for uma gestante.
  - **Pelo menos uma condição deve ser verdadeira**

Significado dos operadores lógicos

Ideia por trás do operador &&

- Você pode obter uma habilitação de motorista se: for aprovado no exame psicotécnico **E** for aprovado no exame de legislação **E** for aprovado no exame de direção.
  - **Todas as condições devem ser verdadeiras**

A	B	A && B
F	F	F
F	V	F
V	F	F
V	V	V

Ideia por trás do operador ||

- Você pode estacionar na vaga especial se: for idoso(a) **OU** for uma pessoa com deficiência **OU** for uma gestante.
  - **Pelo menos uma condição deve ser verdadeira**

A	B	A || B
F	F	F
F	V	V
V	F	V
V	V	V

Significado dos operadores lógicos

Ideia por trás do operador !

- Você tem direito a receber uma bolsa de estudos de você: **NÃO** possuir renda maior que R$ 3000,00.

Significado dos operadores lógicos

Ideia por trás do operador !

- Você tem direito a receber uma bolsa de estudos de você: **NÃO** possuir renda maior que R$ 3000,00.

- Exemplo:
  - João possui renda igual a R$ 5000,00
  - José possui renda igual a R$ 2000,00

  Quem pode ganhar bolsa?

Significado dos operadores lógicos

Ideia por trás do operador !

- Você tem direito a receber uma bolsa de estudos de você: **NÃO** possuir renda maior que R$ 3000,00.

- Exemplo:
  - João possui renda igual a R$ 5000,00
  - José possui renda igual a R$ 2000,00

  Quem pode ganhar bolsa?

- **O operador ! inverte o valor lógico da condição**

Significado dos operadores lógicos

Ideia por trás do operador !

- Você tem direito a receber uma bolsa de estudos de você: **NÃO** possuir renda maior que R$ 3000,00.

- Exemplo:
  - João possui renda igual a R$ 5000,00
  - José possui renda igual a R$ 2000,00

  Quem pode ganhar bolsa?

- **O operador ! inverte o valor lógico da condição**

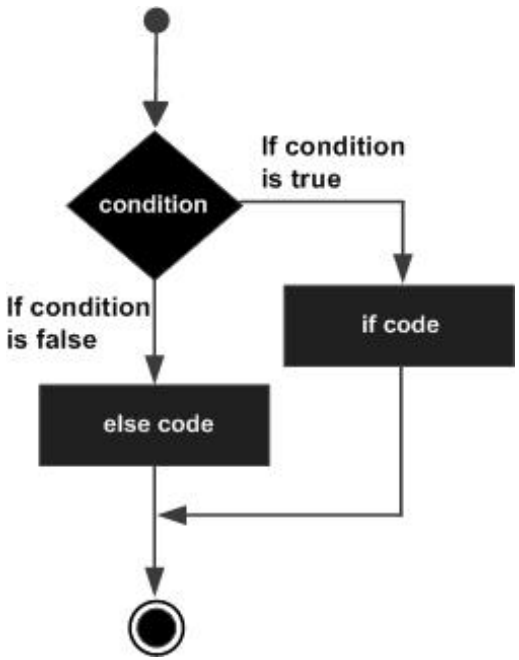
A	!A
F	V
V	F

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Estruturas de controle de fluxo

Estruturas Condicionais — If .. else

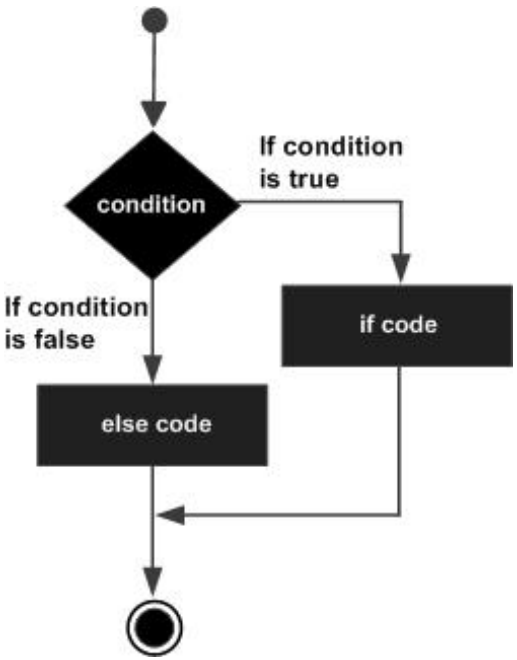
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ




```
1 int c = 67;
2
3 if(c == 1) { // if
4     System.out.println("igual a 1");
5 }
```

Estruturas Condicionais — If .. else

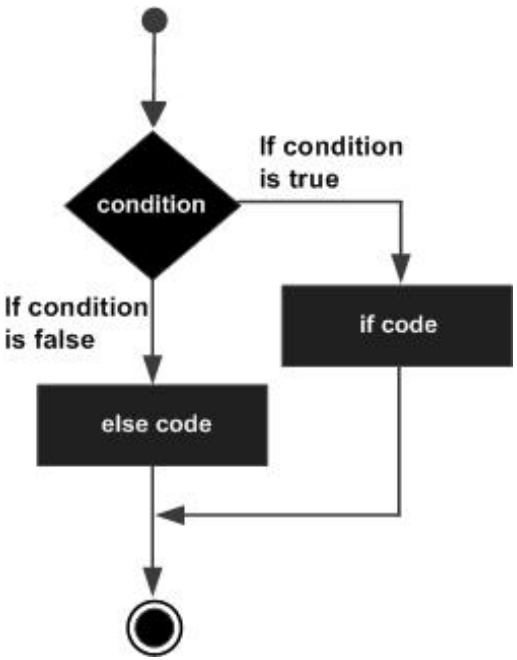
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ




```
1  int c = 67;
2
3  if(c == 1) { // if
4      System.out.println("igual a 1");
5  }
6
7  if(c == 2) { // if .. else
8      System.out.println("igual a 2");
9  } else {
10     System.out.println("nao eh 1 nem 2");
11 }
```

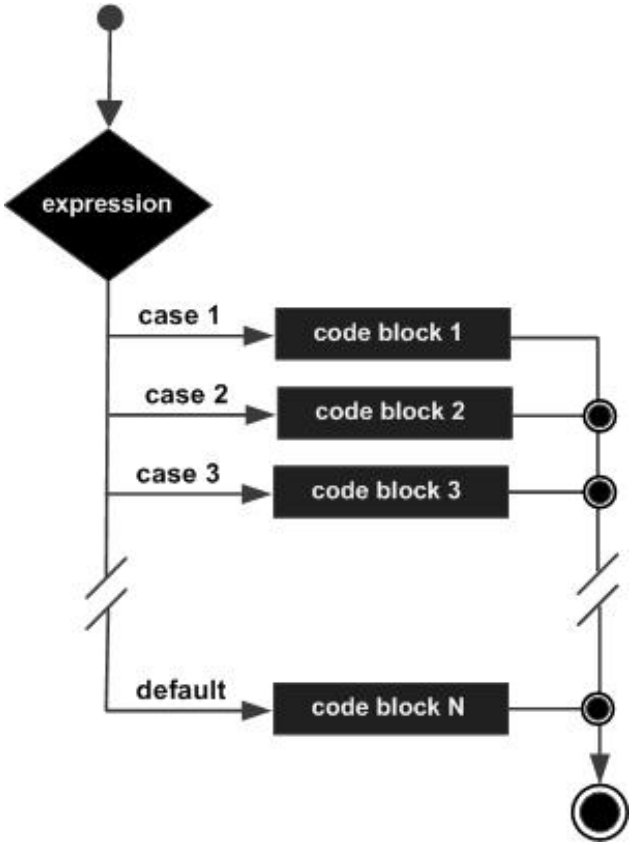
Estruturas Condicionais — If .. else

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



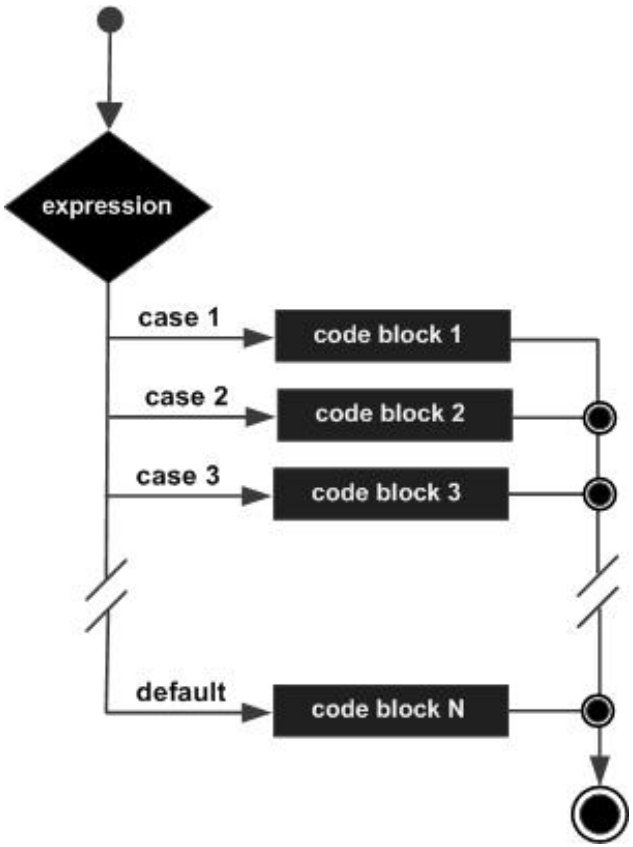

```
1  int c = 67;
2
3  if(c == 1) { // if
4      System.out.println("igual a 1");
5  }
6
7  if(c == 2) { // if .. else
8      System.out.println("igual a 2");
9  } else {
10     System.out.println("nao eh 1 nem 2");
11 }
12
13 if(c == 3) { // if .. else if .. else
14     System.out.println("igual a 3");
15 } else if(c == 4) {
16     System.out.println("igual a 4");
17 } else {
18     System.out.println("nao eh 1,2,3,4");
19 }
```

Estruturas Condicionais — Switch



Estruturas Condicionais — Switch

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ




```
1  Scanner in;
2  in = new Scanner(System.in);
3  int opcao = in.nextInt();
4
5  switch (opcao) // char,byte,short
6  {              // int,String,enum
7      case 1:
8          // comandos aqui
9          break;
10     case 2:
11         // comandos aqui
12         break;
13     case 3:
14         // comandos aqui
15         break;
16     default:
17         // comandos aqui
18         break;
19 }
```

Estrutura condicional ternária

Estrutura opcional ao if-else quando se deseja decidir um **VALOR** com base em uma condição.

Sintaxe:

(condição) ? valor_se_verdadeiro : valor_se_falso

Exemplos:

x = (2 > 4) ? 50 : 80; $\Longrightarrow$ x = 80

str = (10 != 3) ? “Maria” : “Alex”; $\Longrightarrow$ str = “Maria”

Demo


```
1 double preco = 34.5;
2 double desconto;
3 if(preco < 20.0) {
4     desconto = preco * 0.1;
5 }
6 else {
7     desconto = preco * 0.2;
8 }
```

Demo


```
1 double preco = 34.5;
2 double desconto;
3 if(preco < 20.0) {
4     desconto = preco * 0.1;
5 }
6 else {
7     desconto = preco * 0.2;
8 }
```

```
1 double preco;
2 double desconto = (preco < 20.0) ? preco * 0.1 : preco * 0.2;
```

Operadores de atribuição cumulativa

Problema Exemplo

Uma operadora de telefonia cobra R$ 50.00 por um plano básico que dá direito a 100 minutos de telefone. Cada minuto que exceder a franquia de 100 minutos custa R$ 2.00. Fazer um programa para ler a quantidade de minutos que uma pessoa consumiu, daí mostrar o valor a ser pago.

Entrada	Saída
22	Valor a pagar: R$ 50.00

Entrada	Saída
103	Valor a pagar: R$ 56.00

Solução

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  import java.util.Locale;
2  import java.util.Scanner;
3
4  class Programa01 {
5      public static void main(String[] args) {
6          Locale.setDefault(Locale.US);
7          Scanner input = new Scanner(System.in);
8
9          int minutos = input.nextInt();
10
11         double conta = 50.00;
12         if(minutos > 100) {
13             conta = conta + (minutos - 100) * 2.0;
14         }
15
16         System.out.printf("Valor da conta = R$ %.2f%n", conta);
17
18         input.close();
19     }
20 }
```

Operadores de atribuição cumulativa

Universidade Federal do Ceará
Campus Quixadá

Quando se necessita realizar uma operação de uma variável com ela própria, acumulando seu valor, basta utilizar:
<**variável**> <**operador**> **=** <**operando**>

Por exemplo:
int num;
num += 5; *(corresponde a **num = num + 5;**)*
num /= 8; *(corresponde a **num = num / 8;**)*

a += b;	a = a + b;
a -= b;	a = a - b;
a *= b;	a = a * b;
a /= b;	a = a / b;
a %= b;	a = a % b;

Solução com operador cumulativo

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  import java.util.Locale;
2  import java.util.Scanner;
3
4  class Programa02 {
5      public static void main(String[] args) {
6          Locale.setDefault(Locale.US);
7          Scanner input = new Scanner(System.in);
8
9          int minutos = input.nextInt();
10
11         double conta = 50.00;
12         if(minutos > 100) {
13             conta += (minutos - 100) * 2.0;
14         }
15
16         System.out.printf("Valor da conta = R$ %.2f%n", conta);
17
18         input.close();
19     }
20 }
```

Operadores de atribuição cumulativa

Os símbolos $++$ e $--$ são utilizados para incrementar e decrementar em 1 o valor de uma variável numérica.

Sintaxe 1: $++$<**variável**>**;**

- Primeiro incrementa a variável e depois retorna o seu valor.

Sintaxe 2: <**variável**> $--$**;**

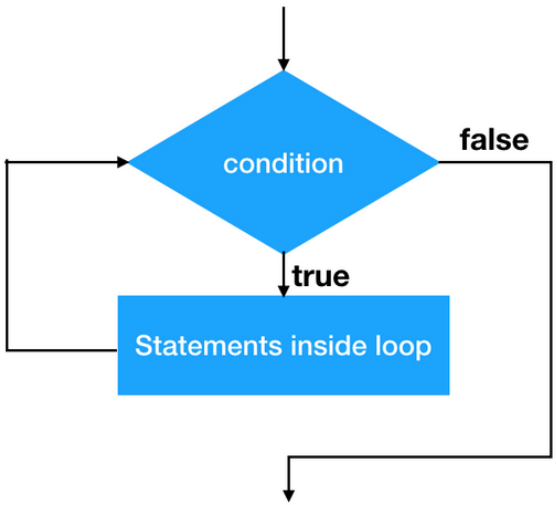
- Primeiro devolve o valor da variável e depois incrementa o seu valor.

Estruturas de repetição

Estrutura de repetição — `while`

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

`while loop`

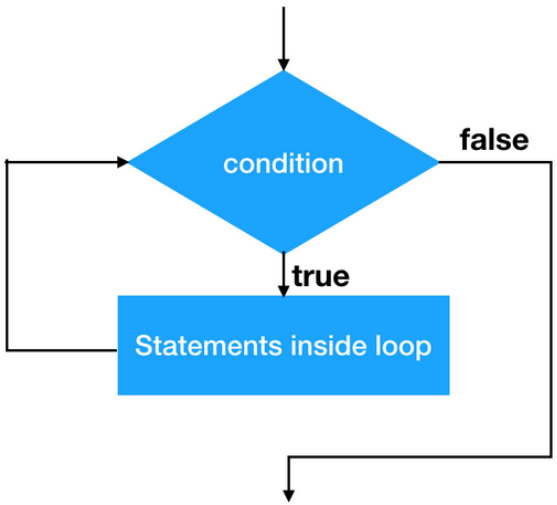



```
1 int contador = 0;
2
3 while (contador < 10) {
4   System.out.println(contador);
5   contador++;
6 }
```

Estrutura de repetição — `while`

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

`while loop`

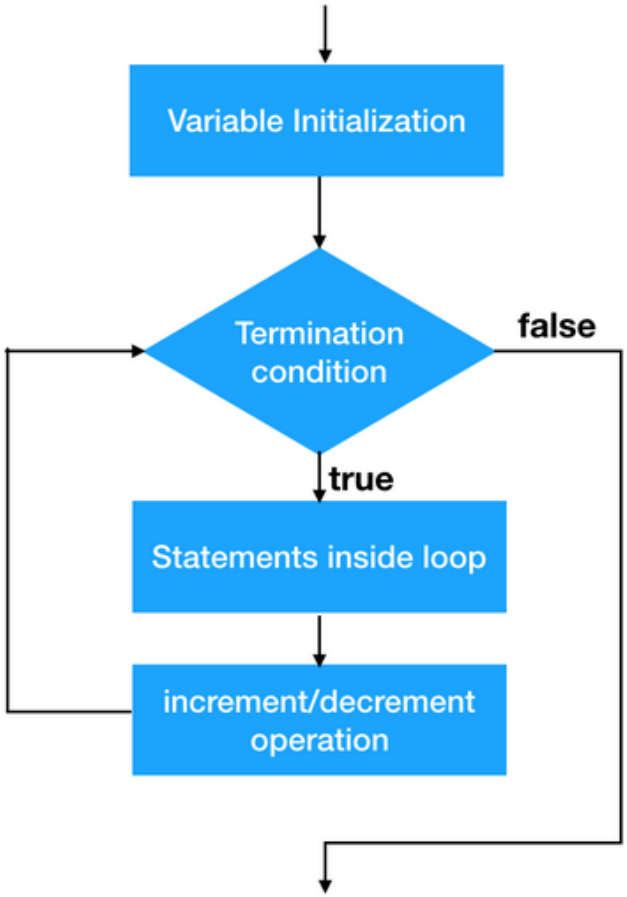



```
1 int contador = 0;
2
3 while (contador < 10) {
4   System.out.println(contador);
5   contador++;
6 }
```

Problema Exemplo:

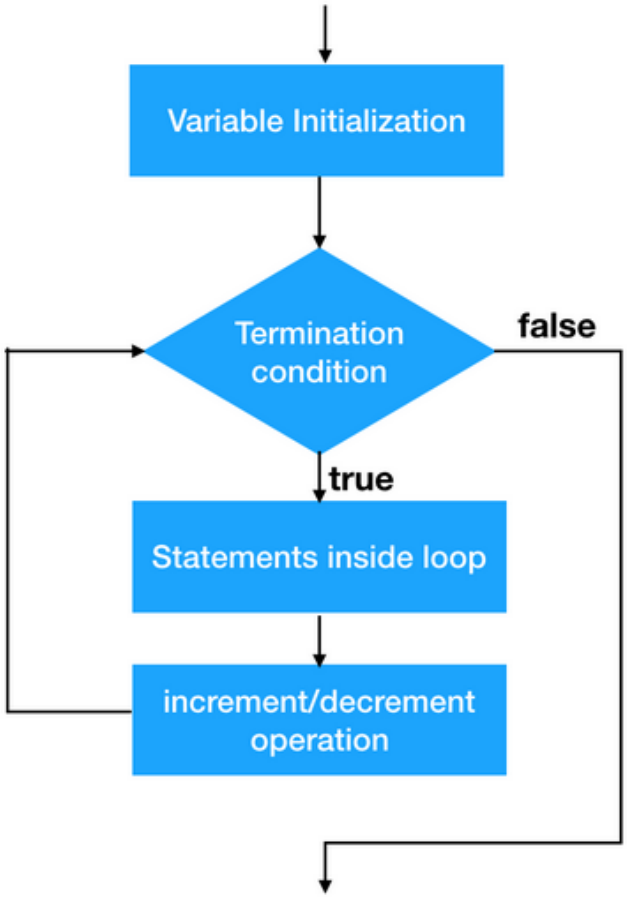
Fazer um programa que lê números inteiros até que um zero seja lido. Ao final mostra a soma dos números lidos.

Estrutura de repetição — `for`




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

Estrutura de repetição — `for`




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

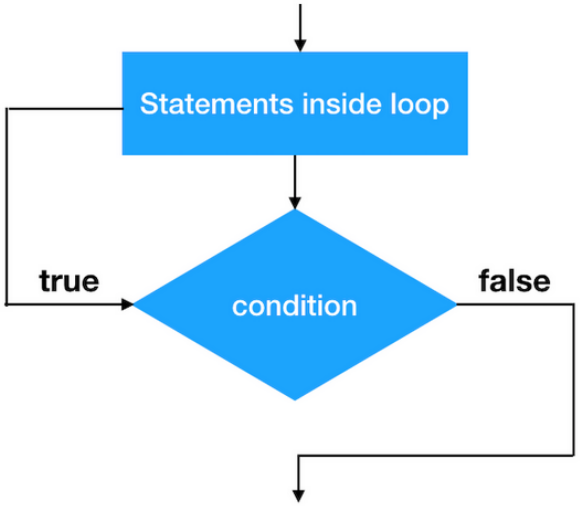
Problema Exemplo:

Fazer um programa que lê um valor inteiro N e depois N números inteiros. Ao final, mostra a soma dos N números lidos

Estruturas de repetição — do..while

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

do..while loop

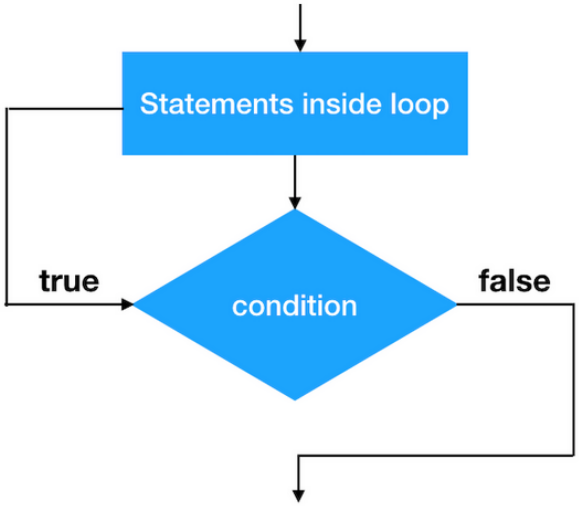



```
1 do {
2   System.out.println(contador);
3   contador++;
4 } while (contador < 10);
```

Estruturas de repetição — do..while

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

do..while loop




```
1 do {
2   System.out.println(contador);
3   contador++;
4 } while (contador < 10);
```

Problema Exemplo:

Fazer um programa para ler uma temperatura em Celsius e mostrar o equivalente em Fahrenheit. Perguntar se o usuário deseja repetir (s/n). Caso o usuário digite "s", repetir o programa.

Fórmula: $F = \frac{9C}{5} + 32$

Funções

Funções (métodos)

- Representam um processamento que possui um significado
  - System.out.println(string)

- Principais vantagens: modularização, delegação e reaproveitamento

- Dados de entrada e saída
  - Funções podem receber dados de entrada (parâmetros ou argumentos)
  - Funções podem ou não retornar um valor

- Em orientação a objetos, funções em classes recebem o nome de métodos.

Problema exemplo

- Escreva um programa para ler três números inteiros e mostrar na tela o maior deles.

Arrays

Arrays

- Um array é uma estrutura de dados que armazena uma coleção de valores do mesmo tipo.

Arrays

- Um array é uma estrutura de dados que armazena uma coleção de valores do mesmo tipo.
- Em Java, arrays são objetos.
  - Os elementos de um array podem ser de tipos nativos (int, float, double, etc.) ou de tipos por referência (objetos).

Arrays

- Um array é uma estrutura de dados que armazena uma coleção de valores do mesmo tipo.
- Em Java, arrays são objetos.
  - Os elementos de um array podem ser de tipos nativos (int, float, double, etc.) ou de tipos por referência (objetos).
- Suponha que queremos armazenar 100 inteiros no nosso programa. Como declarar um array para armazenar esses 100 inteiros?

Arrays

- Um array é uma estrutura de dados que armazena uma coleção de valores do mesmo tipo.
- Em Java, arrays são objetos.
  - Os elementos de um array podem ser de tipos nativos (int, float, double, etc.) ou de tipos por referência (objetos).
- Suponha que queremos armazenar 100 inteiros no nosso programa. Como declarar um array para armazenar esses 100 inteiros?

Sintaxe da declaração de um array: <tipo> [ ] <nome da variável>;

Arrays

- Um array é uma estrutura de dados que armazena uma coleção de valores do mesmo tipo.
- Em Java, arrays são objetos.
  - Os elementos de um array podem ser de tipos nativos (int, float, double, etc.) ou de tipos por referência (objetos).
- Suponha que queremos armazenar 100 inteiros no nosso programa. Como declarar um array para armazenar esses 100 inteiros?

Sintaxe da declaração de um array: <tipo> [ ] <nome da variável>;

Exemplo: `int[] arr;`

Qual valor está armazenado na variável `arr`?

Arrays

- Um array pode ser inicializado usando o operador `new`:

```
int[] arr = new int[100];
```

Arrays

- Um array pode ser inicializado usando o operador `new`:

```
int[] arr = new int[100];
```

- O valor que determina o tamanho do array deve ser um inteiro:

```
int n = 100;
int[] arr = new int[n];
```

Arrays

- Um array pode ser inicializado usando o operador `new`:

```
int[] arr = new int[100];
```

- O valor que determina o tamanho do array deve ser um inteiro:

```
int n = 100;
int[] arr = new int[n];
```

- Para acessar um elemento específico no array, usamos o nome da referência para o array e o índice da posição do elemento no array.
  - Em Java, os elementos de um array de tamanho n são numerados de 0 a $n-1$, assim como em C/C++.

Arrays

- Um array pode ser inicializado usando o operador `new`:

```
int[] arr = new int[100];
```

- O valor que determina o tamanho do array deve ser um inteiro:

```
int n = 100;
int[] arr = new int[n];
```

- Para acessar um elemento específico no array, usamos o nome da referência para o array e o índice da posição do elemento no array.
  - Em Java, os elementos de um array de tamanho n são numerados de 0 a $n-1$, assim como em C/C++.

- Todo array sabe o seu tamanho e o armazena na variável **length**, que é um **atributo** do objeto array. A expressão `arr.length` devolve o tamanho do array `arr`.

Exemplo: criando e inicializando um array

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  public class Programa13 {
2      public static void main(String[] args) {
3          // Declarando array e inicializando
4          int[] array = new int[10];
5
6          System.out.printf("%s%8s%n", "Indice", "Valor");
7
8          for(int c = 0; c < array.length; c++) {
9              System.out.printf("%6d%8d%n", c, array[c]);
10         }
11     }
12 }
```

Exemplo: usando uma lista inicializadora

- Podemos criar um array e inicializar seus elementos com uma lista inicializadora, que é uma lista de elementos separados por vírgula e fechada com colchetes.

  Exemplo: `int[] array = {10, 20, 30, 40, 50};`

```
1  public class Programa14 {
2    public static void main(String[] args) {
3      // Declarando array e inicializando
4      int[] array = {32, 27, 64, 18, 95, 14, 90, 70, 60, 37};
5
6      System.out.printf("%s%8s%n", "Indice", "Valor");
7
8      for(int c = 0; c < array.length; c++) {
9        System.out.printf("%6d%8d%n", c, array[c]);
10     }
11   }
12 }
```

O laço `for each`

- O Java possui uma estrutura de repetição (laço) que permite iterar sobre os elementos de um array sem usar uma variável contadora, o for each.

  Sintaxe:

```
for (<Tipo> variavel :  nomeDoArray) {
    ..comandos..
}
```

O laço `for each`

- O Java possui uma estrutura de repetição (laço) que permite iterar sobre os elementos de um array sem usar uma variável contadora, o for each.

Sintaxe:

```
for (<Tipo> variavel :  nomeDoArray) {
    ..comandos..
}
```

Exemplo:

```
int[] array = {5, 78, 23, 10};
for (int element :  array) {
    System.out.println(element);
}
```

O laço `for each`

- O Java possui uma estrutura de repetição (laço) que permite iterar sobre os elementos de um array sem usar uma variável contadora, o for each.

  Sintaxe:

```
for (<Tipo> variavel :  nomeDoArray) {
    ..comandos..
}
```

  Exemplo:

```
int[] array = {5, 78, 23, 10};
for (int element :  array) {
    System.out.println(element);
}
```

- O for each só pode ser usado para obter o valor dos elementos do array, ele não pode modificá-los. Se seu programa precisar iterar e modificar o valor dos elementos, use o for tradicional ou outro laço.

Exemplo: laço for each

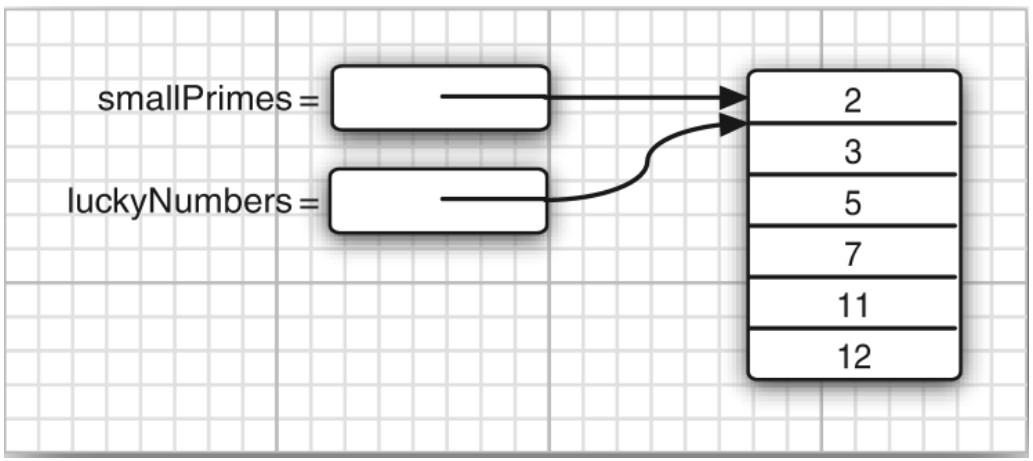

```
1  public class Programa15 {
2      public static void main(String[] args) {
3          int[] array = {87, 78, 90, 45, 23, 17, 67};
4          int total = 0;
5
6          for(int number : array) {
7              total += number;
8          }
9
10         System.out.printf("Total: %d%n", total);
11     }
12 }
```

Cópia entre dois arrays

- Em Java, podemos copiar uma variável array em outra. Porém, após fazer isso, ambas as variáveis passam a referenciar o mesmo array.

  Exemplo:

```
int[] smallPrimes = {2, 3, 5, 7, 11, 13};
int[] luckyNumbers = smallPrimes;
luckyNumbers[5] = 12;
```



UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

FIM