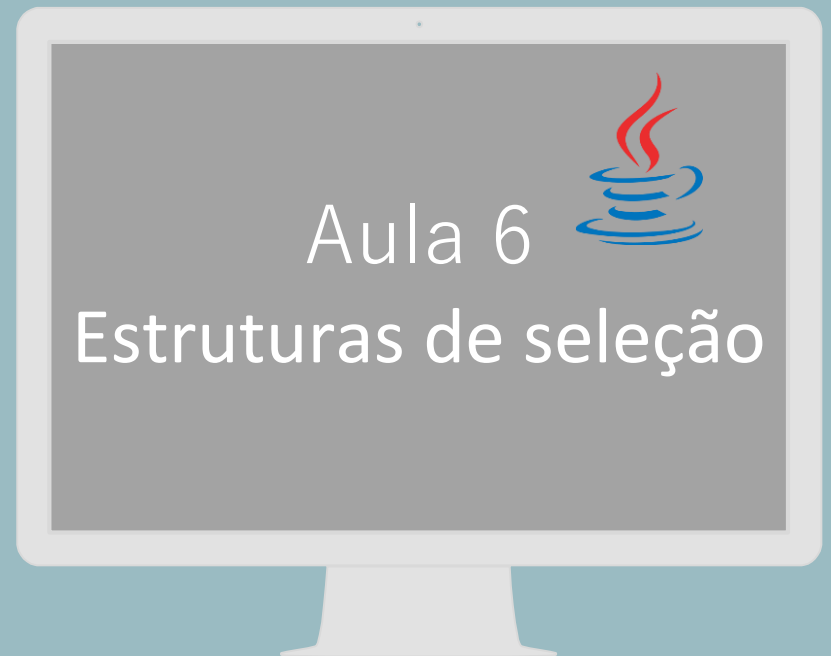




# CURSO DE PROGRAMAÇÃO EM JAVA





*As estruturas de decisão são utilizadas para controlar o fluxo de execução dos aplicativos, possibilitando que a leitura das instruções siga caminhos alternativos em função da análise de determinadas condições. Com elas, é possível condicionar a leitura de uma instrução ou de um bloco delas a uma ou mais condições que precisam ser satisfeitas.*

1.

Instrução if

## Instrução if

A estrutura de decisão if é utilizada para impor uma ou mais condições que deverão ser satisfeitas para a execução de uma instrução ou bloco de instruções. A sua forma geral é a seguinte:

If (<condição>) <instrução ou bloco>

A condição deve ser uma expressão **booleana** que resulte em um valor true ou false. A instrução ou o bloco de instruções somente será executado caso o resultado dessa expressão seja true. Caso o resultado seja false, o fluxo de execução será desviado.



# Instrução if



```
if (expressaoBooleana) {  
    //instruções que serão executadas caso a  
    expressãoBooleana resulte true.  
}
```

```
int hora = 20;  
boolean eManha = false;  
  
// Exemplo 1: com bloco.  
if (hora <= 12) {  
    eManha = true;  
    System.out.print(hora + " AM");  
}  
  
//Exemplo 2: sem bloco.  
if (!eManha)  
    System.out.print(hora - 12 + " PM");  
  
System.out.println(" é o mesmo que " + hora + " horas.");  
//Esta linha é incondicionalmente exibida
```

2.

Instrução if-else

## Instrução if-else

A estrutura de decisão if-else é uma variação da estrutura if. Ela é utilizada para impor uma ou mais condições que deverão ser satisfeitas para a execução de uma instrução ou bloco de instruções e possibilita a definição de uma instrução ou bloco de instruções a serem executados caso as condições não sejam satisfeitas. A sua forma geral é a seguinte:

If(<Condição>)      <instrução      ou  
bloco>

else <instrução ou bloco>



# Instrução if-else



```
| if (expressaoBooleana) {  
|     //instruções que serão executadas caso a  
| expressaoBooleana resulte true.  
| } else {  
|     //instruções que serão executadas caso a  
| expressaoBooleana resulte false.  
| }
```

```
| int hora = 20;  
|  
| if (hora <= 12)  
|     System.out.print(hora + " AM");  
| else  
|     System.out.print(hora - 12 + " PM");  
|  
| System.out.println(" é o mesmo que " + hora + " horas.");  
| //Esta linha é incondicionalmente exibida  
|
```



3.

Instruções if...else  
aninhadas

## Instruções if...else aninhadas

As instruções **if** ou **if...else** podem ser aninhadas dentro de outras instruções **if** ou **if...else** para casos em que antes de determinadas instruções serem executadas sejam necessárias combinações de resultados de expressões booleanas.



# Estrutura if-else



```
if (expressaoBooleana1) {  
    if (expressaoBooleana2) {  
        // Instruções a serem executadas caso as expressões  
            booleanas 1 e 2 resultem em true.  
    } else {  
        // Instruções a serem executadas caso a expressão  
            booleana 1 resulte em true, e a 2 em false.  
    }  
} else {  
    if (expressaoBooleana3) {  
        // Instruções a serem executadas caso a expressão  
            booleana 1 resulte em false, e a 2 em true.  
    } else {  
        // Instruções a serem executadas caso as expressões  
            booleanas 1 e 3 resultem em false.  
    }  
}
```

# Estrutura if-else



```
int hora = 20;

if (hora < 0 || hora >= 24)
    if (hora < 0)
        System.out.print("A hora deve ser maior que 0.");
    else
        System.out.print("A hora deve ser menor que 24.");
else {
    if (hora <= 12)
        System.out.print(hora + " AM");
    else
        System.out.print(hora - 12 + " PM");

    System.out.println("é o mesmo que " + hora + " horas.");
}
```

## Instruções if...else aninhadas

É possível verificar no exemplo anterior que a primeira instrução if mesmo contendo mais de uma linha de instruções consegue identificar que o if...else forma uma única ramificação e assim executar a expressão booleana normalmente. Isso se deve ao fato que toda else está vinculada a uma if.

Já no else com o escopo mais externo, verifica-se chaves delimitadoras de bloco. Essas chaves são necessárias por conta de uma segunda instrução, nomeadamente `System.out.println()`.



# Estrutura if-else



```
int hora = 20;

if (hora < 0)
    System.out.print("Erro: A hora deve ser maior que 0.");
else if (hora >= 24)
    System.out.print("Erro: A hora deve ser menor que 24.");
else if (hora <= 12)
    System.out.print(hora + " AM é o mesmo que " + hora + "
horas.");
else
    System.out.print((hora - 12) + " PM é o mesmo que " +
hora + " horas.");
```



## Instruções if...else aninhadas

No exemplo acima há um recurso estilístico para indentar o código com a finalidade de aprimorar a legibilidade. As palavras chave **if** foram anexadas às palavras chave **else** já que as **else** têm somente uma instrução cada e por isso não necessitam de chaves para delimitar bloco de instruções. O próximo código tem exatamente a mesma funcionalidade apesar das quebras de linha.



# Estrutura if-else



```
int hora = 20;

if (hora < 0)
    System.out.print("Erro: A hora deve ser maior que 0.");
else
    if (hora >= 24)
        System.out.print("Erro: A hora deve ser menor que 24.");
    else
        if (hora <= 12)
            System.out.print(hora + " AM é o mesmo que " + hora + " horas.");
        else
            System.out.print(hora + " PM é o mesmo que " + hora + " horas.");
```



4.

Instrução switch

## Instrução switch


A estrutura de decisão **switch** é uma forma simples para se definir diversos desvios no código a partir de uma única variável ou expressão.

Havendo uma variável com diversos valores possíveis e sendo necessário um **tratamento específico para cada um** deles, o uso da estrutura if-else se torna confuso e dificulta a leitura do código. Nesse caso, a clareza e a facilidade estão do lado da estrutura switch.



# Instrução switch

Dentro do parâmetro da **switch** pode ser utilizada expressão que resulte em: **byte**, **short**, **char**, **int** e **String**. As chaves que delimitam o bloco são necessárias ainda que só haja uma ramificação do fluxo do código.



```
switch (expressao) {  
    case constante1:  
        // Instruções  
        break;  
    case constante2:  
        // Instruções  
        break;  
    default:  
        // Instruções  
}
```

A palavra chave **case** indica as ramificações de código. Deve ser seguida de uma expressão constante que **corresponda ao tipo** da expressão inserida no parâmetro da **switch**, e essa expressão constante, por sua vez, deve ser seguida de **:** que é o carácter que delimita o início do bloco de instruções relativo à **case**. Após **:** podem ser inseridas 0 ou mais instruções, incluindo a palavra chave **break** que será abordada mais adiante. Ao iniciar outra instrução **case** ou inserir a chave de fechamento do bloco de **switch** o bloco anterior é encerrado.

# Instrução switch



```
int dia = 5;
final int segunda = 2;
final int sexta = 6;
switch (dia) {
    case segunda:
        System.out.print("Segunda ");
    case 3:
        System.out.print("Terça ");
    case 4:
        System.out.print("Quarta ");
    case 5:
        System.out.print("Quinta ");
    case sexta:
        System.out.print("Sexta ");
    case 7:
        System.out.print("Sábado ");
    case 0:
    case 1:
        System.out.print("Domingo ");
}
```

## Instrução **break**

Caso seja necessário que apenas sejam executadas instruções vinculadas a determinadas **case** então deve-se utilizar a instrução **break**. Após a instrução **break** o fluxo do programa sai do bloco de **switch**



# Instrução break



```
switch (dia) {  
    case segunda:  
        System.out.print("Segunda ");  
    case 3:  
        System.out.print("Terça ");  
    case 4:  
        System.out.print("Quarta ");  
    case 5:  
        System.out.print("Quinta ");  
    case sexta:  
        System.out.print("Sexta ");  
        break;  
    case 7:  
        System.out.print("Sábado ");  
    case 0:  
    case 1:  
        System.out.print("Domingo ");  
}  
System.out.println("\n-> Fora do bloco de instruções de  
switch.");
```



## Instrução **default**

A instrução **default** pode ser utilizada para o caso da expressão no parâmetro de **switch** não corresponder a nenhum dos valores das instruções **case**.

**Default** pode aparecer em qualquer ordem e segue o mesmo funcionamento que **case** no que tange a bloco de instruções e uso de **break**.



# Instrução default



```
switch (dia) {  
    case segunda:  
        System.out.print("Segunda ");  
    case 3:  
        System.out.print("Terça ");  
    case 4:  
        System.out.print("Quarta ");  
    case 5:  
        System.out.print("Quinta ");  
    case sexta:  
        System.out.print("Sexta ");  
        break;  
    case 7:  
        System.out.print("Sábado ");  
    case 1:  
        System.out.print("Domingo ");  
    default:  
        System.out.print("Dia inválido");  
}
```



# Exercício teste



```
int a = 5;

if (a > 2)
    if (a < 4)
        System.out.println("a é igual a 3.");
else
    System.out.println("a é menor ou igual a 2.");
```

O que será impresso na tela? A afirmação está correta?

# Exercício teste



```
int a = 5;

if (a > 2)
    if (a < 4)
        System.out.println("a é igual a 3.");
else
    System.out.println("a é menor ou igual a 2.");
```

O que será impresso na tela? A afirmação está correta?

Será exibido a é **menor que 2.** porém a informação está incorreta. A instrução else está vinculada a if imediatamente anterior, e no parâmetro dessa **if** a expressão booleana resultará **false**. Logo, a informação correta seria **a é maior ou igual a 4..**

# Exercício teste



```
int a = 5;

if (a > 2)
    if (a < 4)
        System.out.println("a é igual a 3.");
else
    System.out.println("a é menor ou igual a 2.");
```

Como corrigir o código?

```
int a = 5;

if (a > 2) {
    if (a < 4)
        System.out.println("a é igual a 3.");
} else
    System.out.println("a é menor ou igual a 2.");
```

# Exercício teste



Resposta:

```
double b = 10.5;
double c = 2.0;

if (b == c) {
    System.out.println("b é igual a c");
} else {
    if (b > c) {
        System.out.println("b é maior que c");
    } else {
        System.out.println("b é menor que c");
    }
}
```

A última expressão booleana ( $b < c$ ) não é necessária pois caso um número  $b$  não seja maior ou igual a um número  $c$ , obviamente ele será menor.

4.

# Operador Ternário

# Operador Ternário

É um operador matemático, com um **condicional**, ele necessita de **três variáveis**. Lembrando também que o ternário sempre deve retornar valor, e o valor será sempre do mesmo tipo, para ambos os lados da expressão.

Boolean ? var1(true) : var2(false);



# Operador Ternário



```
int valor1 = 1;
int valor2 = 2;

System.out.println(valor1 > valor2 ? "1 é maior" : "2 é
maior"));

if(valor1 > valor2){
    System.out.println("1 é maior");
} else {
    System.out.println("2 é maior");
}
```

# Exercício teste



```
int valor1 = 20;  
int valor2 = 15;  
int valorEscolhido;
```

```
valorEscolhido = valor1 > valor2 ?valor1: valor2;
```

```
System.out.println("O Valor escolhido é = " +  
valorEscolhido);
```

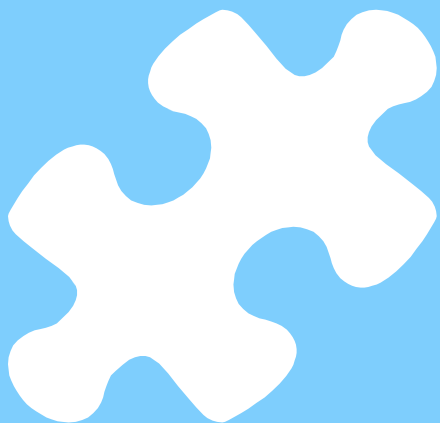
```
valor1 = 10;  
valorEscolhido = valor1 > valor2 ?valor1: valor2;
```

```
System.out.println("O Valor escolhido é = " +  
valorEscolhido);
```

O Valor escolhido é = 20

O Valor escolhido é = 15





# DESAFIO

E aí, vamos praticar?

# Intervalo

Você deve fazer um programa que leia um valor qualquer e apresente uma mensagem dizendo em qual dos seguintes intervalos ( $[0,25]$ ,  $(25,50]$ ,  $(50,75]$ ,  $(75,100]$ ) este valor se encontra. Obviamente se o valor não estiver em nenhum destes intervalos, deverá ser impressa a mensagem "Fora de intervalo".

O símbolo  $($  representa "maior que". Por exemplo:

$[0,25]$  indica valores entre 0 e 25.0000, inclusive eles.

$(25,50]$  indica valores maiores que 25 Ex: 25.00001 até o valor 50.0000000

## Entrada:

25.01

25.00

100.00

-25.02

## Saída:

Intervalo (25,50]

Intervalo [0,25]

Intervalo (75,100]

Fora de intervalo

# Tempo de Jogo

Leia a hora inicial, minuto inicial, hora final e minuto final de um jogo. A seguir calcule a duração do jogo.

Obs: O jogo tem duração mínima de um (1) minuto e duração máxima de 24 horas.

## Entrada:

7 8 9 10

7 7 7 7

7 10 8 9

## Saída:

O JOGO DUROU 2 HORA(S) E 2 MINUTO(S)

O JOGO DUROU 24 HORA(S) E 0 MINUTO(S)

O JOGO DUROU 0 HORA(S) E 59 MINUTO(S)

# Obrigado!

## **Alguma pergunta?**

Você pode me contatar em:  
[ywassef@hotmail.com](mailto:ywassef@hotmail.com)