



Ciência da **Computação**

Prof. Luciano Rodrigo Ferretto



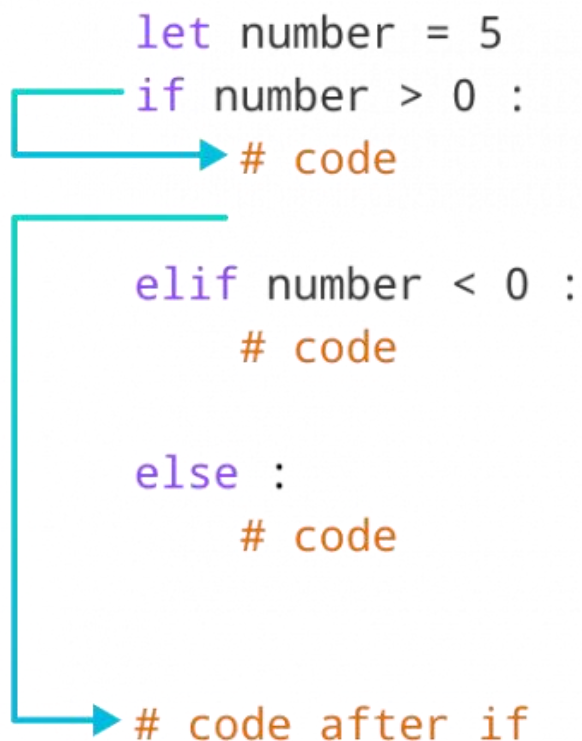
Estruturas Condicionais – Instrução **if**

- Olá, turma! Já que vocês estão familiarizados com a estrutura condicional **if** em Python, será mais fácil compreender a mesma ideia em Java.
- A estrutura **if** é usada para executar um bloco de código **somente se uma condição especificada for verdadeira**. Vamos explorar como isso funciona em Java e como podemos utilizar suas diversas formas e derivações.

Relembrando em Python `if`

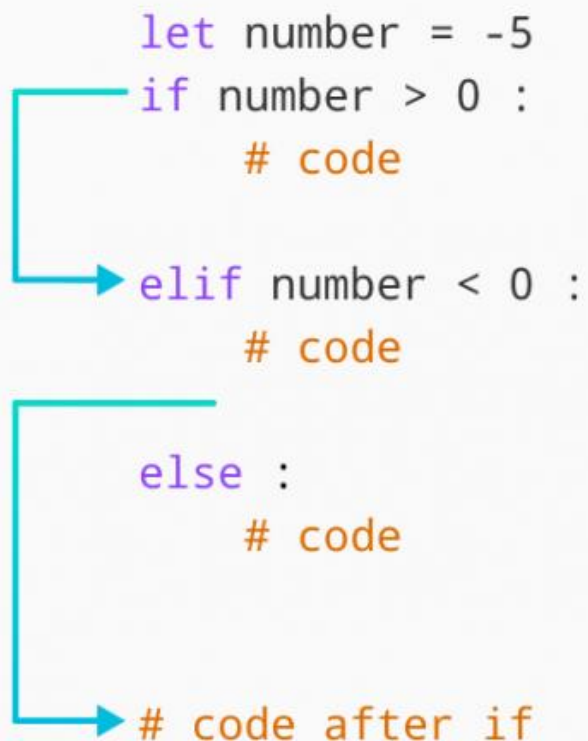
1st Condition is True

```
let number = 5
if number > 0 :
    # code
elif number < 0 :
    # code
else :
    # code
# code after if
```



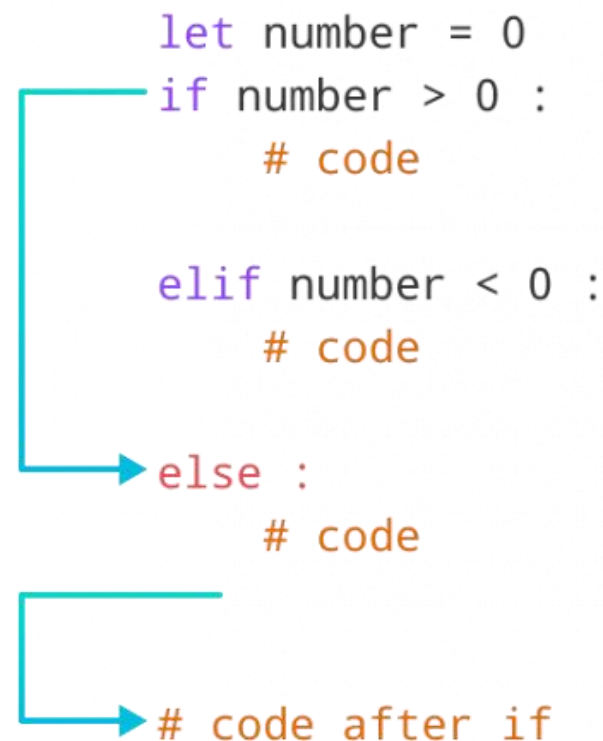
2nd Condition is True

```
let number = -5
if number > 0 :
    # code
elif number < 0 :
    # code
else :
    # code
# code after if
```



All Conditions are False

```
let number = 0
if number > 0 :
    # code
elif number < 0 :
    # code
else :
    # code
# code after if
```





Sintaxe Básica da Instrução **if** em Java

- Assim como no Python, no Java podemos controlar a execução de uma instrução, ou de uma sequência de instruções (bloco), a partir de uma condição específica.
- Quando estamos tratando uma sequência de instruções, estas devem estar entre **chaves {}**.
- Quando é uma linha de instrução somente, não há a necessidade das **chaves {}**, porém podem ser utilizadas também.

```
if(condição)  
{  
    sequência de instruções  
}  
else  
{  
    sequência de instruções  
}
```

```
if(condição) instrução;  
else instrução;
```



Exemplo da Instrução **if** em Java

- Vamos imaginar que queremos construir um algoritmo que receba duas notas de um aluno, e apresente o resultado conforme a média aritmética das notas.



```
1 import java.util.Scanner; //Biblioteca utilizada para acessar os dados digitados pelo usuário.
2 class CalculaMedia { //Nessa classe não vamos realizar o tratamento de exceções.
3
4     public static void main(String args[]){
5         Scanner lerTeclado = new Scanner(System.in); //Inicializa o objeto Scanner passando qual será a
6         //origem dos dados. Neste caso será a entrada no terminal
7         //Solicita ao usuário que informe suas
8         System.out.println("Informe o valor da primeira Nota:");
9         double nota01 = lerTeclado.nextDouble();
10        System.out.println("Informe o valor da segunda Nota:");
11        double nota02 = lerTeclado.nextDouble();
12        lerTeclado.close();
13        //Calcula a média aritmética
14        double mediaAritimetica = (nota01 + nota02) / 2;
15        System.out.printf("A média aritmética é %.2f\n%n", mediaAritimetica);
16
17        if (mediaAritimetica ≥ 7) { //exemplo utilizando com chaves {}
18            System.out.println("PARABÉNS!!! Você foi aprovado");
19        } else //como é apenas uma instrução, também pode ser utilizada sem as chaves {}
20            System.out.println("Infelizmente você não alcançou a média. :-(");
21    }
22 }
```



```
//Exemplo utilizando a estrutura if comum
if (mediaAritimetica ≥ 7) { // com o uso chaves {}
    System.out.println("PARABÉNS!!! Você foi aprovado");
} else { // sem o uso das chaves {}
    System.out.println("Infelizmente você não alcançou a média :-(");
}
```




Instrução **if** aninhado

Um *if aninhado* é uma instrução **if** que é alvo de outro **if** ou **else**. Os **ifs** aninhados são muito comuns em programação. O importante a lembrar sobre **ifs** aninhados em Java é o fato de que uma instrução **else** será sempre referente à instrução **if** mais próxima que estiver dentro do mesmo bloco e ainda não estiver associada a um **else**. Aqui está um exemplo:

```
if (i == 10) {  
    if (j < 20) a = b;  
    if (k > 100) c = d;  
    else a = c; // esse else é referente a if (k > 100)  
}  
else a = d; // esse else é referente a if (i == 10)
```




```
18 ..... if (mediaAritimetica ≥ 7)
19 .....     System.out.println("PARABÉNS!!! Você foi aprovado");
20 ..... else {
21 .....     if (mediaAritimetica ≥ 3) {
22 .....         System.out.println("Infelizmente você deverá realizar o Exame :-(");
23 .....     } else
24 .....         System.out.println("Infelizmente você foi Reprovado! :["");
25 ..... }
```



Instrução **if** aninhado – Escada if-else-if

- Um método bastante utilizado de If aninhado em programação é a *escada if-else-if* (o “*elif*” do python)

```
if (condição)
    instrução;
else if (condição) {
    sequencia_instruções;
} else if (condição)
    instrução;
else
    instrução;
```



```
18 ..... if (mediaAritimetica ≥ 7)
19 .....     System.out.println("PARABÉNS!!! Você foi aprovado");
20 ..... else if (mediaAritimetica ≥ 3)
21 .....     System.out.println("Infelizmente você deverá realizar o Exame :-(");
22 ..... else
23 .....     System.out.println("Infelizmente você foi Reprovado! :^(");
24
```



Operador Ternário

- Java também suporta o **operador ternário** (**? :**), que permite criar estruturas condicionais de forma mais concisa.

condição ? se_true : se_false;

```
int numero = 5;  
String resultado = (numero % 2 == 0) ? "par" : "ímpar";  
System.out.println("O número é " + resultado);
```



Instrução Switch

- A instrução **switch** fornece uma ramificação com vários caminhos.
- Semelhante às instruções de *ifs* aninhados.
- A **expressão** é um valor ou uma expressão cujo resultado é comparado com os diferentes **case**.
- Os **case** são rótulos que representam os diferentes valores possíveis da expressão.
- O bloco de código associado a cada **case** é executado quando a expressão é igual ao valor do **case**.
- O **break** é usado para sair do bloco de código do **switch** após um **case** ser executado. Isso evita a execução dos **case** seguintes.

```
switch(expressão) {  
    case constante1:  
        sequência de instruções  
        break;  
    case constante2:  
        sequência de instruções  
        break;  
    case constante3:  
        sequência de instruções  
        break;  
    .  
    .  
    .  
    default:  
        sequência de instruções  
}
```



Instrução Switch

- As expressões podem ser do tipo **byte**, **short**, **int** e **char**. A partir do Java 7 também pode ser do tipo **String**.
- Cada valor de **case** deve ser um valor literal constante e de tipo compatível com a **expressão**.
- Os valores das constantes **não podem repetir** dentro do mesmo switch.
- As instruções da constante **default** são executadas quando nenhuma das outras constantes coincidem com a expressão.
- O **default** é opcional.

```
switch(expressão) {  
    case constante1:  
        sequência de instruções  
        break;  
    case constante2:  
        sequência de instruções  
        break;  
    case constante3:  
        sequência de instruções  
        break;  
    .  
    .  
    .  
    default:  
        sequência de instruções  
}
```





```
1 import java.util.Random; //Biblioteca utilizada para gerar números aleatórios
2
3 public class ExemploSwitch {
4     public static void main(String[] args) {
5         Random gerador = new Random();
6         int limit = 10;
7         int numeroAleatorio = gerador.nextInt(limit);
8         switch (numeroAleatorio) {
9             case 0:
10                System.out.println("O valor gerado é igual a ZERO");
11                break;
12             case 1:
13                System.out.println("O valor gerado é igual a UM");
14                break;
15             case 2:
16                System.out.println("O valor gerado é igual a DOIS");
17                break;
18             case 3:
19                System.out.println("O valor gerado é igual a TRÊS");
20                break;
21             case 4:
22                System.out.println("O valor gerado é igual a QUATRO");
23                break;
24             default:
25                System.out.println("O valor gerado é maior ou igual a CINCO");
26                break;
27         }
28     }
29 }
```




(JDK 7) Instrução Switch com Strings

```
switch (day) {  
    case "Monday":  
        System.out.println("Week day");  
        break;  
    case "Tuesday":  
        System.out.println("Week day");  
        break;  
    case "Wednesday":  
        System.out.println("Week day");  
        break;  
    case "Thursday":  
        System.out.println("Week day");  
        break;  
    case "Friday":  
        System.out.println("Week day");  
        break;  
    case "Saturday":  
        System.out.println("Weekend");  
        break;  
    case "Sunday":  
        System.out.println("Weekend");  
        break;  
    default:  
        System.out.println("Unknown");  
}
```

(JDK 12) Instrução Switch com Retorno



```
public static void main(String[] args) {  
    String day = "Tuesday";  
    System.out.println(getTipoDia(day));  
}  
  
static String getTipoDia(String day) {  
    return switch (day) {  
        case "Monday":  
            yield "Weekday";  
        case "Tuesday":  
            yield "Weekday";  
        case "Wednesday":  
            yield "Weekday";  
        case "Thursday":  
            yield "Weekday";  
        case "Friday":  
            yield "Weekday";  
        case "Saturday":  
            yield "Weekend";  
        case "Sunday":  
            yield "Weekend";  
        default:  
            yield "Unknown";  
    };  
}
```

(JDK 12) Instrução Switch com Retorno com arrow



```
public static void main(String[] args) {  
    String day = "Tuesday";  
    System.out.println(getTipoDia(day));  
}  
static String getTipoDia(String day) {  
    return switch (day) {  
        case "Monday" → "Week day";  
        case "Tuesday" → "Week day";  
        case "Wednesday" → "Week day";  
        case "Thursday" → "Week day";  
        case "Friday" → "Week day";  
        case "Saturday" → "Weekend";  
        case "Sunday" → "Weekend";  
        default → "Unknown";  
    };  
}
```



(JDK 12) Instrução Switch com Múltiplos valores em case

```
public static void main(String[] args) {  
    String day = "Tuesday";  
    System.out.println(getTipoDia(day));  
}  
  
static String getTipoDia(String day) {  
    return switch (day) {  
        case "Monday", "Tuesday", "Wednesday", "Thursday", "Friday" → "Week day";  
        case "Saturday", "Sunday" → "Weekend";  
        default → "Unknown";  
    };  
}
```



Agora é com Vocês ...

Atividade Avaliativa:

- Escreva um algoritmo que receba do usuário via teclado (Scanner + System.in) o **nome**, **gênero**, **altura** e **peso**
- Após, faça a classificação conforme a tabela no próximo slide:

- Observações:

- Neste nosso exemplo vamos solicitar que o usuário digite o gênero com as seguintes possibilidades:
 - 'M' para Masculino
 - 'F' para Feminino
 - 'N' para aqueles que não desejarem informar (para esses deverá ser usado o padrão Feminino da tabela)
- Cálculo IMC = $\text{Peso} / (\text{Altura}^2)$
- A Classe **Scanner** não possui um método para receber um **char**, dessa forma utilize o método **next()** e extraia somente o primeiro caractere com o método **charAt()** da classe String conforme o exemplo abaixo:
 - `genero = lerTeclado.next().charAt(0);`

IMC



Classificação	Masculino	Feminino
Obesidade Mórbida	Maior ou igual 40	Maior ou igual 39
Obesidade Moderada	30 a 39,9	29 a 38,9
Obesidade Leve	25 a 29,9	24 a 28,9
Normal	20 a 24,9	19 a 23,9
Abaixo do Normal	Menos de 20	Menos de 19



Utilizem o recurso Text Block disponível como um recurso padrão desde a versão Java 15

```
String textBlock = """
    Aqui temos um exemplo de Bloco de Texto
    --- Você poderá imprimir essa variável 'textBlock'
    --- Será mostrado na tela, da forma como está aqui no código
    --- Inclusive com a Identação
    FICA A DICA GALERA!!!!
    """;
System.out.println(textBlock);
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
Aqui temos um exemplo de Bloco de Texto
    --- Você poderá imprimir essa variável 'textBlock'
    --- Será mostrado na tela, da forma como está aqui no código
    --- Inclusive com a Identação
    FICA A DICA GALERA!!!!
```




Limpar a tela



Embora o Java não possua um método nativo para limpar o terminal, existem **diversas maneiras de alcançar esse objetivo**, cada uma com suas vantagens e desvantagens.

Imprimindo caracteres de escape:

Utilize a sequência de escape `\033[H\033[2J` dentro de uma String e imprima na tela:

```
System.out.print("\033[H\033[2J");
```

Vantagens:

Portabilidade: Funciona em diferentes sistemas operacionais, pois utiliza caracteres de escape ANSI.

Desvantagens:

Menos intuitivo: A sequência de escape pode parecer complexa para alguns usuários.

```
... System.out.print("\033[H\033[2J");
```



O problema do buffer do teclado



Quando você utiliza o método **scan.nextDouble()**, o Scanner lê e retira o próximo número de ponto flutuante do buffer de entrada do teclado.

No entanto, o Scanner **deixa** o caractere de **quebra de linha (\n)** no buffer, que foi digitado junto com o número.

Consequências:

Se você tentar utilizar outro método do Scanner imediatamente após o `scan.nextDouble()`, sem consumir o caractere de quebra de linha, ele interpretará esse caractere como parte da próxima entrada, causando erros ou comportamentos inesperados.

Solução: `scan.nextLine()`

O método `scan.nextLine()` consome toda a linha restante do buffer de entrada, incluindo o caractere de quebra de linha. Isso garante que o buffer esteja limpo e pronto para receber a próxima entrada corretamente.

```
Scanner scan = new Scanner(System.in);

System.out.print("Digite um número: ");
double num = scan.nextDouble();
scan.nextLine();
// Consome o \n e a linha completa
System.out.print("Digite seu nome: ");
String nome = scan.nextLine();
```



Leitura de caractere único



A classe Scanner da biblioteca padrão do Java **não** possui um método específico chamado **nextChar()** para ler um único caractere do teclado.

Por que não existe um nextChar?

Foco em tokens: O Scanner foi projetado para ler tokens completos (palavras, números, etc.) do teclado, facilitando a leitura de dados estruturados.

Leitura de caracteres: Se você precisa ler um único caractere você pode pegar o primeiro token através do **next()**, ou toda a linha através do **nextLine()**, e a partir da String retornada, retirar o primeiro caractere através do método **charAt(int index)**

Lembre-se: os métodos **next()** e **nextLine()** retornam uma String, e uma String é um **objeto** que possui seus próprios atributos e métodos, como no exemplo o **charAt**

```
System.out.println("Digite seu gênero");
String genero = ler.nextLine();
char generoChar = genero.charAt(0);
// ou
char _generoChar = ler.nextLine().charAt(0);
```