

# Java - Aula 04

## Estruturas de Controle

---

**Cristiano Amaral Maffort**

`cristiano@cefetmg.br`

`maffort@gmail.com`

Técnico em Informática

Departamento de Computação

CEFET-MG – Belo Horizonte



# Agenda

- Comando de decisão
- Comando de seleção
- Comandos de repetição

# Instrução `if`

- A forma básica da instrução `if` é

`if` (condição-lógica)

`instrução;`

`[else`

`instrução;]`

# Instrução `if`

- No caso de existir mais de uma instrução tanto em `if` quanto em `else`
  - devem ser usadas chaves (`{ }`) para delimitar os blocos de instruções.

```
if (condição-lógica) {  
    sequência de instruções;  
}  
  
[else {  
    sequência de instruções;  
}]
```

# ifs em sequência

```
[...]
```

```
if (i == 10)
```

```
    a = b;
```

```
if (j < 15)
```

```
    c = d;
```

```
else if (j > 15)
```

```
    b = d;
```

```
else
```

```
    a = d;
```

```
[...]
```

# ifs aninhados

```
[...]
if (i == 10) {
    if (j < 20)
        a = b;
}
else
    a = d;
[...]
```

# ifs aninhados

```
[...]  
if (i == 10) {  
    if (j < 20)  
        a = b;  
    if (k > 100)  
        c = d;  
    else  
        a = c;  
}  
else  
    a = d;  
[...]
```

# Instrução `switch`

- Fornece uma ramificação de vários caminhos
  - fazendo uma seleção, em várias alternativas, a partir de uma mesma expressão.

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



# Instrução `switch`

- A expressão que aciona a seleção alternativa deve ser dos tipos
  - `byte`, `short`, `int`, `char`, `String` ou uma enumeração
  - Dos tipos integrais, `long` não é um tipo válido para a expressão

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

# if ou switch

- Quando usar `ifs` sequenciais no lugar de `switch`?

# if OU switch

- Quando usar `ifs` sequenciais no lugar de `switch`?
  - Quanto a condição de seleção (expressão) depender de diferentes valores

```
int j, k, d;  
[...]  
if (j < 15) // ...  
else if (k != 0) // ...  
else if (d > 0) // ...  
else // ...
```

# if OU switch

- Quando usar `ifs` sequenciais no lugar de `switch`?
  - Quando o tipo da expressão não for compatível com a instrução `switch`

```
[long, double, float] j;
```

```
[...]
```

```
if (j == 0) // ...
```

```
else if (j == 1) // ...
```

```
else if (j == 2) // ...
```

```
else // ...
```

# if OU switch

- Quando usar `ifs` sequenciais no lugar de `switch`?
  - Quanto a condição de seleção (expressão) possuir conjuntos contíguos de valores (classes de equivalências sequenciais extensas)

```
int j;  
[...]  
if (j > 100) // ...  
else if (j > 50) // ...  
else if (j > 20) // ...  
else // ...
```

# Instrução `while`

- A forma básica da instrução `while` é  
`while` (condição-continuidade)  
    instrução;

- **OU**

```
while (condição-continuidade) {  
    instruções;  
}
```

# Instrução do-while

- A forma básica da instrução do-while é

```
do {  
    instruções;  
} while (condição-continuidade);
```

# Instrução `for`

- A forma básica da instrução `for` é

```
for (inicialização; condição-continuidade; iteração)  
    instrução;
```

- **ou**

```
for (inicialização; condição-continuidade; iteração) {  
    instruções;  
}
```



# Variantes da Instrução `for`

- `for (i = 0, j = 10; i < j; i++, j -= 2)`
- `for ( ; (char)System.in.read() != 'S'; )`
- `for (i = 0; i < 10; )`
- `for ( ; ; ) // loop infinito?`
- `for (int i = 1; i <= 5; fatorial *= i, i++); // sem corpo`
- `List<String> funcionarios = new ArrayList<String>();`  
    `for (String funcionario: funcionarios) [ ... ]`

# Instrução break

- Força a saída imediata de um laço (qualquer laço),
  - Ignorando o código restante em seu corpo/bloco.
- O fluxo de execução continua na primeira instrução externa ao laço.

```
public static void main(String[] args) {  
  
    Scanner scan = new Scanner(System.in);  
    for ( ; ; ) {  
        char cmd = scan.next().charAt(0);  
  
        if (cmd == 'q')  
            break;  
  
        System.out.println(": " + cmd);  
    }  
  
    System.out.println("Fim do programa...");  
}
```

# Instrução continue

- Força uma iteração antecipada de um laço (qualquer laço),
  - Ignorando o código restante em seu corpo/bloco.
- O fluxo de execução continua na iteração e verificação de continuidade do laço.

```
public static void main(String[] args) {  
  
    Scanner scan = new Scanner(System.in);  
    int n = scan.nextInt();  
  
    // números ímpares até n  
    for (int i = 1; i < n; i ++ ) {  
        if (i % 2 == 0)  
            continue;  
  
        System.out.print(i + " ");  
    }  
  
    System.out.print("\nFim do programa...");  
}
```

# Bibliografia Obrigatória

- HORSTMANN, Cay S.; CORNELL, Gary. ***Core Java: Fundamentos***. 8. ed. São Paulo: Pearson Prentice Hall, 2020, p. 19 – 42.
- SCHILDT, Herbert; SKRIEN, Dale. ***Programação com Java: uma introdução abrangente***. Porto Alegre: AMGH, 2013, p. 75 – 115.