

Operador ternário

- Tipo de condicional
- (exp. booleana) ? código V : código F;
 - Código V -> caso a expressão booleana seja verdadeira
 - Código F -> caso a expressão booleana seja falsa
- Retorna um dos códigos
- Retornar = devolver um valor

```
class OperadorTernario{
    public static void main(String[] args) {
        int numeroDias = 16; //valor entre 1 e 30
        String quinzena;
        // Ternário
        quinzena = (numeroDias <= 15) ? "Primeira</pre>
quinzena": "Segunda quinzena";
        System.out.println(quinzena);
        // Equivalente
        if (numeroDias <= 15) {</pre>
            quinzena = "Primeira quinzena";
        } else {
            quinzena = "Segunda quinzena";
        System.out.println(quinzena);
```

```
import java.util.Scanner;
public class Switch {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        char primeiraLetra = teclado.next().charAt(0);
        char segundaLetra = "Gato gordo".charAt(1);
        switch (primeiraLetra) {
            case 'a':
               System.out.println("Letra A de \"Amo Cuscuz\"");
               break;
            case 'b':
               System.out.println("Letra B de \"Bora comer cuscuz?\"");
               break;
            case 'c':
                System.out.println("Letra C de \"Comi Cuscuz\"");
               break;
           default:
               System.out.println("Ueh, o caractere nao eh: A, B ou C");
               break;
```

Switch

- Substitui cadeias de if's e else's;
- "case" equivale ao "if" ou "else if";
- "default" equivale ao "else";
- Fácil substituição das condições;
- Mais limitado.

Laços de repetição

- Laços de repetição, também chamados de estruturas de repetição ou loopings:
 - São estruturas que repetem blocos de códigos com um mesmo padrão;
- Exemplos:
 - Contagem;
 - Inserção de vários dados;
 - Mostra de vários dados:
 - Tabelas;
 - Listas.

```
Exemplo de lógica:

numero = 0

somar +1 em numero

mostrar(numero) -> 1

somar +1 em numero

mostrar(numero) -> 2

somar +1 em numero

mostrar(numero) -> 3
```

```
Exemplo de lógica:
numero = 0
repetir 3x{
 somar +1 em numero
 mostrar(numero)
```

While

- Repete **enquanto** certa condição for VERDADEIRA!!
- While = enquanto.

Saída do código ao lado:

```
PS C:\Users\marli\Dev\Java\tutoria\revisao\codigos-slides02> java While.java
1
2
3
4
5
6
7
8
9
```

```
public class While {
    public static void main(String[] args) {
        int n = 0;
        while (n < 10) {
            n++;
            System.out.println(n);
```

Do-while

- Executa o bloco, depois repete enquanto for verdadeiro!
- Diferença do while:
 - Executa um bloco somente se a condição for verdadeira.
 - Faz (do), depois executa o bloco enquanto (while) a condição for verdadeira.

```
public class DoWhile {
    public static void main(String[] args) {
        int n = 0;
        do {
            n++;
            System.out.println(n);
        } while (n < 0);
    }
}</pre>
```

Saída do código ao lado:

PS C:\Users\marli\Dev\Java\tutoria\revisao\codigos-slides02> **java** DoWhile.java

For

- Estrutura do for:
 - for (inicialização ; condição ; finalização){
 - · }
 - Inicialização: declaração inicial antes de começar a repetição;
 - Condição: expressão que deve ser verdadeira para que a repetição ocorra;
 - Finalização: código que será executado sempre que uma repetição termina.

```
public class For {
   public static void main(String[] args) {
       int n = 10;
       for (int i = 0; i < n; i++) {
            System.out.println("O valor de i eh: "+i);
```

Vetor

- Também chamado de array ou lista;
- São listas (limitadas);
- Variável de variáveis;
- Caixa comprida com várias caixas dentro.

doces



tarefas

0 Acordar
1 Tomar café
2 Se exercitar
3 Ler livros
4 Ser feliz

Vetor

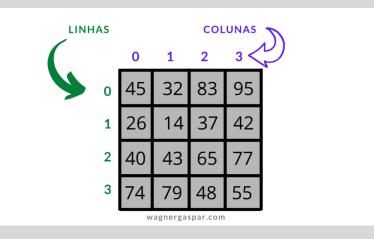
- Estrutura de criação:
- tipo[] nome = new tipo[comprimento];
- ∘ Ex.:
- o int[] numeros = new int[10];
- Para acessar um dos índices do vetor:
- nome[posicao];

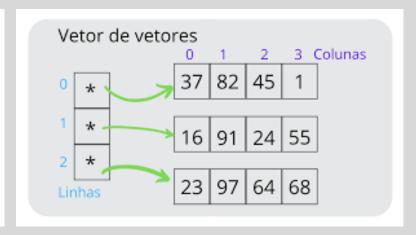
```
class Vetor{
    public static void main(String[] args) {
        String[] tarefas = new String[5];
        tarefas[0] = "Acordar";
        tarefas[1] = "Tomar café";
        tarefas[2] = "Se exercitar";
        tarefas[3] = "Ler livros";
        tarefas[4] = "Ser feliz";
        // Alternativa:
        // String[] tarefas = {"Acordar", "Tomar café",
"Se exercitar", "Ler livros", "Ser feliz"};
        for (int i = 0; i < tarefas.length; i++) {</pre>
            System.out.print(tarefas[i]);
```

Matriz

- Vetor de vetores/array de arrays
- "Tabela"







Matriz

- Estrutura
- Criação:
- tipo[][] nome = new tipo[linhas][colunas];
- Único:
- o nome[0][1];

```
public class Matriz {
   public static void main(String[] args) {
        int[][] tabela = {{2, 15}, {5, 19}};
       for (int i = 0; i < 2; i++) {
           for (int j = 0; j < 2; j++) {
               System.out.print(tabela[i][j]+" ");
            System.out.println();
```

PS C:\Users\marli\Dev\Tutoria\poo-2021\java-basico\codigos\parte02> java Matriz.java 2 15 5 19

Método

- São "ações" em uma classe
- Bloco de código executado somente quando chamado, quando o programador quiser
- As variáveis existentes no método só servem dentro de tal
- Métodos estáticos (static) só podem referenciar "coisas" (métodos e variáveis) estáticas

Saída do código ao lado:

Fora do metodo Dentro do metodo Fora do metodo

```
public class Metodo {
   public static void main(String[] args) {
       String texto = "Fora do metodo";
       System.out.println(texto);
       mostrarAlgo();
       System.out.println(texto);
    static void mostrarAlgo(){
       // System.out.println(texto); ERRO, pois texto não
existe
       System.out.println("Dentro do metodo");
```

Método – Passagem de parâmetro

- Como dito anteriormente, as variáveis dentro de um método só são acessíveis dentro dele
- Mas, para ir contra essa limitação, existe a passagem de parâmetro por valor
- Leva um ou vários valores para dentro do método

Saída do código ao lado:

```
Fora do metodo
Fora do metodo
Fora do metodo
```

```
public class MetodoPassagem {
    public static void main(String[] args) {
        String texto = "Fora do metodo";
        System.out.println(texto);
        mostrarAlgo(texto);
        System.out.println(texto);
    static void mostrarAlgo(String txt){
        System.out.println(txt);
```

Método - Retorno

- Para complementar a passagem de parâmetro, existe o retorno
- Leva um valor para fora do método
- Quando um valor é retornado, a chamada do método age de forma semelhante a uma variável

Saída do código ao lado:

Fora do metodo Dentro do metodo Dentro do metodo Dentro do metodo

```
public class MetodoRetorno {
    public static void main(String[] args) {
        System.out.println("Fora do metodo");
        System.out.println(darValor());
        System.out.println(darValor());
        System.out.println(darValor());
    static String darValor(){
        return "Dentro do metodo";
```

FIM (mas continua)

Agora só nos resta praticar.

Após isso, continuaremos com esses horários separados para vocês...

A PREFERÊNCIA dos horários de revisão será para assuntos anteriores!

Isso NÃO indica que é EXCLUSIVAMENTE de assuntos anteriores!

Referências

• https://wagnergaspar.com/estrutura-de-dado-matriz-array-bidimensional-em-portugol/