<ESTRUTURASBÁSICAS DEJAVA>

{ Estruturas básicas de Java

<Declarando Variáveis>

As variáveis são locais na memória utilizados para armazenar algum valor e para que esse possa ser utilizado e/ou tratado posteriormente. Existem dois tipos de variáveis em Java: os tipos primitivos e as variáveis de referência.

Tipos primitivos

Um tipo de dado primitivo especifica o tamanho e o tipo de valores variáveis, e não tem métodos adicionais.

Tipo de dado	Tamanho	Descrição
byte	1 byte	Armazena números inteiros de -128 a 127
short	2 bytes	Armazena números inteiros de -32768 a 32767
int	4 bytes	Armazena números inteiros de -2147483648 a 2147483647
long	8 bytes	Armazena números inteiros de -9223372036,854775808 a 9223372036854775807
float	4 bytes	Armazena números fracionários. Suficiente para armazenar 6 a 7 dígitos decimais
double	8 bytes	Armazena números fracionários. Suficiente para armazenar 15 dígitos decimais
boolean	1 bit	Armazena valores “true” ou “false”
char	2 bytes	Armazena um único caractere/letra ou valores ASCII

Tipos primitivos não são orientados a objeto, mas devido à sua grande precisão e velocidade de processamento nunca foram retirados do Java.

A declaração de variáveis é feita informando um tipo e seu identificador. Elas devem ser declaradas e inicializadas antes de serem utilizadas, e seu identificador deve ser único dentro de seu escopo.
Ex.:

```
tipo nomeVariavel = valor;
```

O identificador (nome da variável) é formado por caracteres Unicode (padrão que permite aos computadores representar e manipular, de forma consistente, texto de qualquer sistema de escrita existente). Eles podem ser formados por letras, cifrão ($), underline (_) e números. Um identificador não pode iniciar com um número, mas após o primeiro caractere, números podem ser utilizados. Os identificadores precisam ser palavras únicas e nunca duas palavras separadas. Quando for necessário utilizar duas palavras, podemos separá-las com o _ ou usarmos o camelCase, que é iniciar a segunda palavra com letra maiúscula, como vimos no exemplo nomeVariavel.

Caracteres Unicode

Unicode é um padrão que fornece um número único para cada caractere, não importando a plataforma, o programa e nem a linguagem. Por exemplo: os números 65 e 66 se referem às letras A e B (maiúsculas), e os números 97 e 98 às letras a e b (minúsculas). Este padrão tem sido adotado por líderes do setor de informática como a Apple, HP, IBM, Microsoft, Oracle, SAP e muitos outros.

Variáveis finais

Se você não quiser que outros (ou você mesmo) substituam os valores existentes, use a palavra-chave final (isto irá declarar a variável como “final” ou “constante”, o que significa imutável e somente leitura).

<Variáveis de referência>

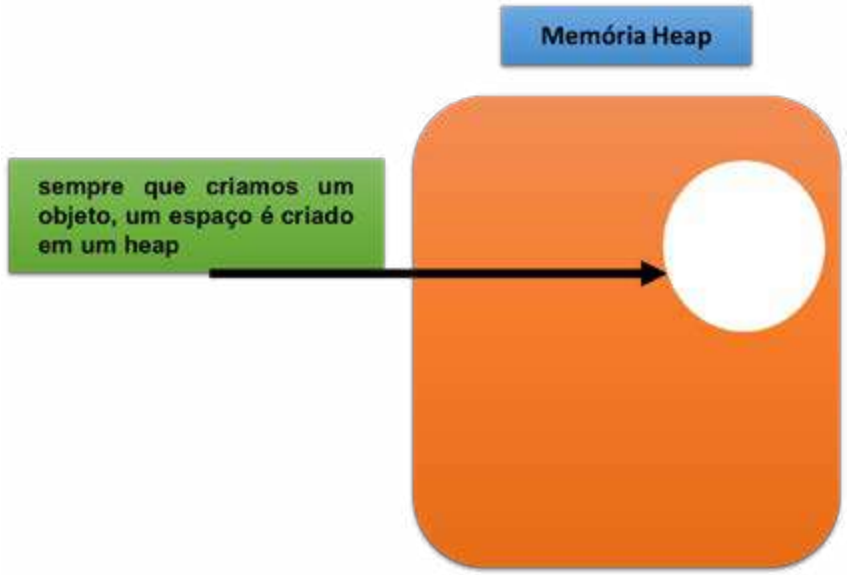
A variável de referência é usada para apontar objetos / valores. Classes, interfaces, matrizes, enumerações e anotações são tipos de referência em Java. Essa variável contém os objetos/valores de tipos de referência em Java.

Quando criamos um objeto (instância) de classe, o espaço é reservado na memória heap (local onde o Java armazena suas variáveis e instâncias de objetos).

Ex.:

```
Demonstração D1 = new demonstração();
```

Para acessar esse espaço, criamos um ponteiro que simplesmente aponta o objeto (o espaço criado em uma Memória Heap).



A variável de referência também pode armazenar valor nulo. Por padrão, se nenhum objeto for passado para uma variável de referência, ela armazenará um valor nulo.

Além disso, é possível acessar membros de objeto usando uma variável de referência ao aplicar sintaxe de ponto.

Ex.:

```
<nome da variável de referência>. <nome da variável de instância / nome do método>
```

Os operadores são utilizados para realizar operações sobre variáveis e valores.

{ Operadores Java

Operador	Nome	Descrição	Exemplo
+	Adição	Soma dois valores	x + y
-	Subtração	Subtrai um valor de outro	x - y
*	Multiplicação	Multiplica dois valores	x * y
/	Divisão	Divide um valor por outro	x / y
%	Módulo	Devolve o restante da divisão	x % y
++	Incremento	Aumenta o valor de uma variável em 1	++x
--	Decremento	Diminui o valor de uma variável em 1	--x

Os operadores de atribuição são usados para atribuir valores a variáveis.

<Operadores de Atribuição Java>

Operador	Exemplo	Igual a
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Os operadores de comparação são usados para comparar dois valores.

<Operadores de Comparação Java>

Operador	Nome	Exemplo
==	igual	x == y
!=	diferente	x != y
>	maior a	x > y
<	menor a	x < y
>=	maior ou igual a	x >= y
<=	menor ou igual a	x <= y

Os operadores lógicos são usados para determinar a lógica entre as variáveis ou valores.

<Operadores Lógicos Java>

Operador	Nome	Descrição	Exemplo
&&	E	Retorna verdadeiro se ambas as afirmações forem verdadeiras.	x < 5 && x < 10
||	OU	Retorna verdadeiro se uma das declarações for verdadeira.	x < 5 || x < 4
!	NÃO	Reverte o resultado. Retorna falso se o resultado for verdadeiro.	!(x < 5 && x < 10)

Em operações com o &&(e), o resultado só será verdadeiro(true) se ambas as condições forem verdadeiras.

Em operações com o ||(ou), é preciso que apenas uma das duas seja verdadeira para que retorne verdadeiro(true).

Em operações com !(não), a condição é negada. Ou seja, sempre que colocarmos uma verdadeira, se essa estiver precedida de !, ela passa a ser falsa; ou se for falsa, e tiver precedida do !, passa a ser verdadeira.

Existe a tabela da verdade, tanto para o && como para o ||, que auxiliam no entendimento:

&& e

Ex.: Se for de noite e estiver calor, vou beber um suco(true).

A	B	A && B
verdadeiro	verdadeiro	verdadeiro
verdadeiro	falso	falso
falso	verdadeiro	falso
falso	falso	falso

Ex.: Se for de noite ou estiver calor, vou beber um suco(true).

A	B	A && B
verdadeiro	verdadeiro	verdadeiro
verdadeiro	falso	verdadeiro
falso	verdadeiro	verdadeiro
falso	falso	falso

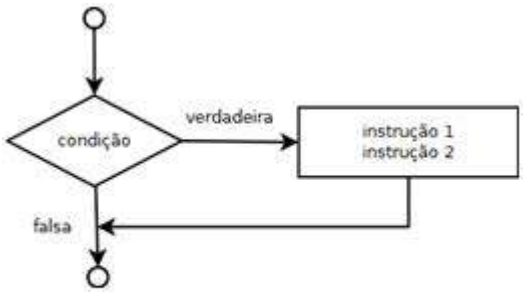
{ Estruturas condicionais e de repetição

Estruturas condicionais e de repetição são fundamentais para a maioria das linguagens de programação. Sem elas, as instruções dos programas seriam executadas sequencialmente sem nenhum tipo de reaproveitamento ou ramificação.

Essa estrutura deve avaliar uma condição, e somente se esta condição for verdadeira, duas instruções

<Estruturas Condicionais>

serão executadas como parte do fluxo principal.



<If...else>

A declaração If é utilizada para especificar um bloco de código a ser executado se uma condição for verdadeira, e a declaração else se a condição for falsa.

```
if (condition) {
  // block of code to be executed if the
condition is true
} else {
  // block of code to be executed if the
condition is false
}
```

Além disso, é possível utilizar a estrutura else if para especificar uma nova condição se a primeira condição for falsa.

```
if (condition1) {
 // block of code to be executed if
condition1 is true
} else if (condition2) {
// block of code to be executed if the
condition1 is false and condition2 is true
} else {
// block of code to be executed if the
condition1 is false and condition2 is false
}
```

<Switch/case>

A estrutura condicional switch/case vem como alternativa em momentos em que temos que utilizar múltiplos ifs no código. Múltiplos if/else encadeados tendem a tornar o código muito extenso, pouco legível e com baixo índice de manutenção.

O switch/case testa o valor contido em uma variável, realizando uma comparação com cada uma das opções. Cada uma dessas possíveis opções é delimitada pela instrução case.

A análise de cada caso também precisa ter seu final delimitado e essa delimitação é feita através da palavra break.

```
switch(expression) {
    case x:
    // code block
       break;
    case y:
    // code block
       break;
    default:
    // code block
}
```

<Estruturas de repetição>

Estruturas de repetição, também conhecidas como loops (laços), são utilizadas para executar repetidamente uma instrução ou bloco de instrução enquanto determinada condição estiver sendo satisfeita.

As principais estruturas de repetição na maioria das linguagens são o for e o while.

<For>

O for é uma estrutura de repetição. Seu ciclo será executado por um tempo ou condição pré-determinados e em uma quantidade de vezes que determinamos.

```
for (<variável de controle>, <análise da variável de
controle>, <incremento da variável de controle>) {
// Código a ser executado
}
```

<While>

O while também é uma estrutura de repetição, assim como o for. A diferença entre ambas é que, enquanto se utiliza o for quando geralmente conhecemos a quantidade de vezes que o trecho de código deverá ser repetido, o while é utilizado quando não sabemos exatamente quantas vezes o código será repetido.

```
while (<condição>) {
// Trecho de código a ser repetido
}
```

Leitura complementar:

For-each

Para cada uma é uma outra técnica de travessia como para o loop, enquanto o loop.

Em vez de declarar e inicializar uma variável do contador de loop, você declara uma variável que é do mesmo tipo que o tipo base da matriz, seguida por dois pontos, que é então seguida pelo nome da matriz.
No corpo do loop, você pode usar a variável de loop que você criou em vez de usar um elemento de array indexado.
É comumente usada para iterar sobre um array ou uma classe de Coleções (por exempli, ArrayList).

```
for (type var : array) {
statements using var;
}
```

Equivalente a:

```
for (int i=0; i<arr.lenght; i==) {
type var = arr [i];
statements using var;
}
```

<Leituras complementares>

https://www.w3schools.com/java/default.asp

<Referências>

https://www.w3schools.com/java/java_variables.asp

https://acervolima.com/variavel-de-referencia-em-java/

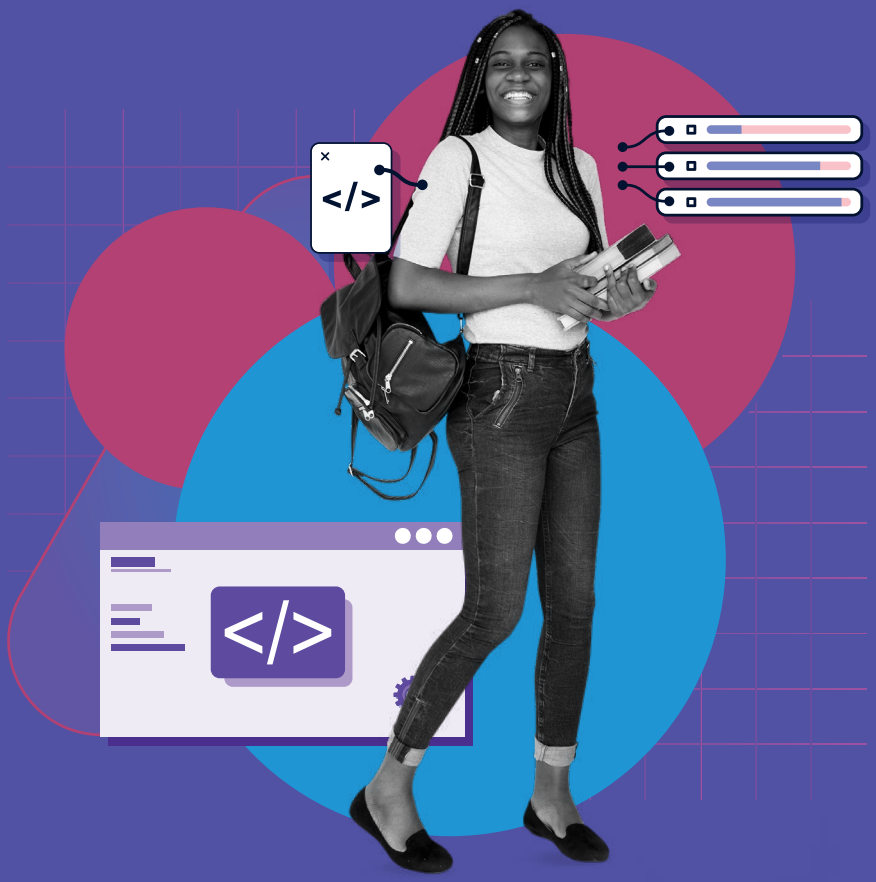
https://www.devmedia.com.br/estruturas-condicionais-em-java/21135

https://www.treinaweb.com.br/blog/estruturas-condicionais-e-estruturas-de-repeticao-em-java

Google olabi

<ESTRUTURASBÁSICAS DEJAVA>

/exercícios/



1 ➔ Escreva um algoritmo que leia um número de ponto flutuante e imprima "zero" se o número for zero. Caso contrário, imprima "positivo" ou "negativo". Adicione "pequeno" se o valor absoluto do número for inferior a 1, ou "grande" se exceder 1.000.000.

2 ➔ Escreva um algoritmo que leia o nome e as três notas obtidas por um aluno durante o semestre. Calcule a média (aritmética), informe o nome e sua menção: Aprovado (média >= 7), Reprovado (média <= 5) e Recuperação (média entre 5.1 a 6.9).

3 ➔ Escreva um algoritmo para resolver equações quadráticas (use se, senão se e de outra forma).

4 ➔ Escreva um programa algoritmo para encontrar o número de dias em um mês.

5 ➔ Leia um número qualquer fornecido pelo usuário. Determine se o número é maior do que 50, imprimindo uma mensagem indicando tal fato.

6 ➔ Dado um número inteiro qualquer, fornecido pelo usuário, descubra se o mesmo é par ou ímpar.

7 ➔ Crie uma variável chamada “fruta”. Essa variável deve receber uma string com o nome de uma fruta. Após, crie uma estrutura condicional switch que receba essa variável e que possua três casos:
Caso maçã: retorne no console “Não vendemos essa fruta aqui”.
Caso kiwi: retorne “Estamos com escassez de kiwis.”
Caso melancia: retorne “Aqui está. São 3 reais o quilo”.
Teste com essas três opções e verifique o console de seu navegador. Crie também um default, que retornará uma mensagem de erro no console.

8 ➔ Elabore um algoritmo que some a idade de 5 pessoas utilizando while.

9 ➔ Imprima a tabuada de um número inserido pelo usuário.

<Referência para os exercícios>

https://www.w3resource.com/java-exercises/

Professor: Julio Cesar | Monitora: Thassia Lima