Fundamentos de Java

Programação Java: Conceitos Iniciais

Softblue
cursos online

Tópicos Abordados

- Variáveis
  - Declaração, inicialização e alteração de valor
- Tipos primitivos do Java
- Operadores
  - Aritméticos
  - Comparação
  - Lógicos
- Casting implícito e explícito
- O tipo de dados *char*

Tópicos Abordados

- Estruturas de controle
  - *if-else*
  - *switch*
  - *while*
  - *do-while*
  - *for*
  - *break*
  - *continue*
- Comentando código em Java

Declaração de Variáveis

- Variáveis devem possuir um tipo e um nome

```
int      anoNasc;
double   peso;
char     sexo;
boolean  canhoto;
```

Tipo de dado | Nome da variável

Inicialização de Variáveis

- Para inicializar variáveis, utilizamos o operador "=" (atribuição)

```
anoNasc = 1980;
peso = 65.7;
sexo = 'M';
```

- É possível também declarar e inicializar simultaneamente

```
double altura = 1.8;
```

- O Java não inicializa as variáveis automaticamente

Alteração do Valor de Variáveis

- Outros exemplos de uso de variáveis

```
int contador = 20;
int novoContador = contador + 1;
```

`novoContador = 21`

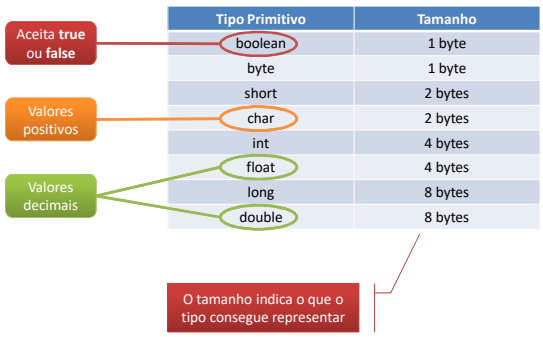
```
int x = 15;
x = x + 1;
```

`x = 16`

```
int y = x + x - 10;
```

`y = 22`

Tipos Primitivos do Java



Tipo Primitivo	Tamanho
boolean	1 byte
byte	1 byte
short	2 bytes
char	2 bytes
int	4 bytes
float	4 bytes
long	8 bytes
double	8 bytes

A variável *var*

- A partir do Java 10 é possível declarar uma variável como *var*
- O tipo que a variável vai assumir vai depender do valor colocado nela

```
var x = 2;
```
➡ int

```
var y = 10.3;
```
➡ double

```
var z = true;
```
➡ boolean

Operadores Aritméticos

Operador	Descrição
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo

Operadores de Comparação

Operador	Descrição
==	Igual
!=	Diferente
<	Menor que
>	Maior que
<=	Menor ou igual a
>=	Maior ou igual a

Operadores Lógicos

Operador	Descrição
!	Negação
||	OU
&&	E

Outros Operadores Importantes

Operador	Descrição	Exemplo
++	Incremento	x++
--	Decremento	x--
+=	Soma com valor e atribui o resultado à própria variável	x += 2
-=	Subtrai do valor e atribui o resultado à própria variável	x -= 5
*=	Multiplica pelo o valor e atribui o resultado à própria variável	x *= 3
/=	Divide pelo valor e atribui o resultado à própria variável	x /= 4

Operadores de Incremento e Decremento

- Os operadores de incremento ("++") e decremento ("--") podem ser de dois tipos
  - Pré-fixados
    - Ex: *++x;*
  - Pós-fixados
    - Ex: *x++;*

```
int x = 10;
int y = ++x;
```
➡ x = 11, y = 11

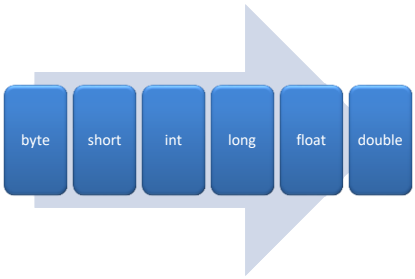
```
int x = 10;
int y = x++;
```
➡ x = 11, y = 10

Casting

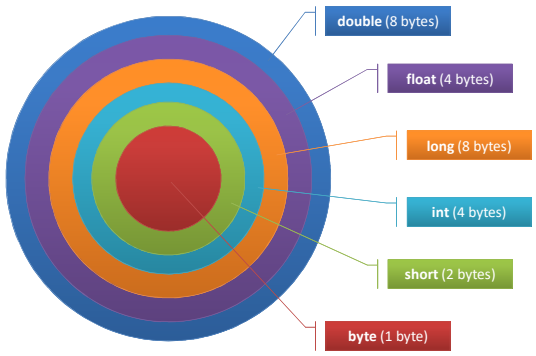
- O casting consiste em atribuir uma variável/valor de um tipo a uma variável de outro tipo
- Podem ser implícitos ou explícitos

Casting Implícito

- O Java faz a conversão do tipo de dado automaticamente



Casting Implícito



Exemplos de Casting Implícito

```
long n1 = 10;
```

10 é do tipo **int** e pode ser atribuído a uma variável **long**

```
float n2 = 5L;
```

5L é do tipo **long** e pode ser atribuído a uma variável **float**

```
double n3 = 2.3f;
```

2.3f é do tipo **float** e pode ser atribuído a uma variável **double**

```
int n4 = 3.5;
```

3.5 é do tipo **double** e não pode ser atribuído a uma variável **int**. É necessário um casting explícito.

Casting Explícito

- A conversão deve ser feita pelo programador

```
double d = 100.0;
int i = d;
```

→

```
double d = 100.0;
int i = (int) d;
```

- Cuidado com o casting explícito!

```
int n1 = (int) 3.5;
```

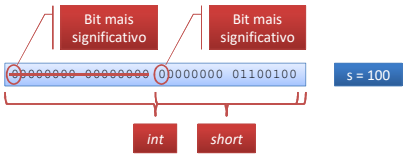
O resultado é **3**.
Como o *int* não armazena a parte decimal, ela é perdida.

```
byte n2 = (byte) 129;
```

O resultado é **-127**.
O número 129 é muito grande para caber dentro de uma variável do tipo *byte*.

Casting Explícito: Exemplo #1

```
int i = 100;
short s = (short) i;
```

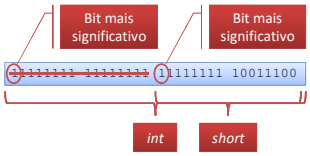


s = 100

O bit mais significativo define o sinal do número: **0** é **positivo**; **1** é **negativo**

Casting Explícito: Exemplo #2

```
int i = -100;
short s = (short) i;
```



11111111 10011100

Complemento

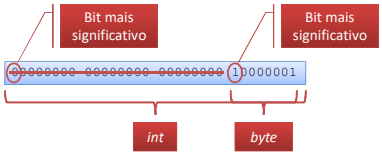
00000000 01100011

+1

00000000 01100100 → 100 → s = -100

Casting Explícito: Exemplo #3

```
int i = 129;
byte b = (byte) i;
```



10000001

Complemento

01111110

+1

01111111 → 127 → b = -127

O Tipo de Dados *char*

- O *char* é o único tipo primitivo em Java sem sinal
- Um *char* indica um caractere, sendo utilizadas aspas simples na sua representação

```
char c = 'A';
```

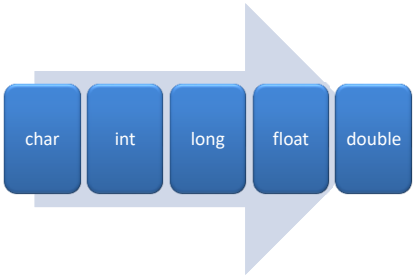
- A atribuição de números a um *char* também é válida

```
char c = 65;
```

Código ASCII do 'A'

O Tipo de Dados *char*

- O cast implícito ocorre a partir do tipo *int*



Estruturas de Controle: *if-else*

- Sintaxe básica

```
if (<condição_booleana>) {
    <código_se_condição_verdadeira>;
}
```

- Opcionalmente, pode existir uma cláusula *else*

```
if (<condição_booleana>) {
    <código_se_condição_verdadeira>;
} else {
    <código_se_condição_falsa>;
}
```

Estruturas de Controle: *if-else*

- A condição booleana pode ser qualquer expressão cujo resultado seja *true* ou *false*

```
int x = 50;

if (x > 30) {
    System.out.println("Número maior que 30");
} else {
    System.out.println("Número menor que 30");
}
```

Estruturas de Controle: *if-else*

- Outra possibilidade é utilizar o **operador ternário** para substituir o *if-else*

```
int x = 50;
boolean r;

if (x > 30) {
    r = true;
} else {
    r = false;
}
```

```
int x = 50;
boolean r;

r = x > 30 ? true : false;
```

Resultado, se verdadeiro

Resultado, se falso

Estruturas de Controle: *switch*

- A estrutura *switch* funciona de forma semelhante a um *if-else*

```
int i = 1;

switch (i) {
    case 1:
        System.out.println("Valor = 1");
        break;
    case 2:
        System.out.println("Valor = 2");
        break;
    default:
        System.out.println("Valor não reconhecido");
}
```

Estruturas de Controle: *switch*

- A expressão avaliada pelo *switch* deve ser
  - Um valor que possa ser convertido para *int*
  - Um elemento de um enum
  - Uma *String*
- Caso o código entre num bloco *case* que não possua *break*, todos os *cases* abaixo serão executados até que um *break* seja encontrado
  - Nesta situação, inclusive o bloco *default* é executado
- O bloco *default* é semelhante ao bloco *else*

Estruturas de Controle: *while*

- Repete determinado código enquanto uma condição for verdadeira
- A condição é testada no início do bloco

```
int idade = 15;

while (idade < 18) {
    System.out.println(idade);
    idade = idade + 1;
}
```

Estruturas de Controle: *do-while*

- Semelhante ao *while*
- A condição é testada no fim do bloco

```
int contador = 10;
do {
    System.out.println(contador);
    contador = contador + 1;
} while (contador < 20);
```

Estruturas de Controle: *for*

- Semelhante ao *while*, mas possui seção para declaração de variáveis para o loop

```
for (int i = 0; i < 10; i++) {
    System.out.println(i);
}
```

```
for (;;) {
    System.out.println("loop infinito");
}
```

Estruturas de Controle: *break*

- Permite forçar a saída de um loop

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break;
    }
    System.out.println(i);
}
```

Estruturas de Controle: *continue*

- Força o loop a executar o próximo passo

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        continue;
    }
    System.out.println(i);
}
```

Comentando Código em Java

- Comentários em uma linha (//)

```
//exemplo de comentário no código em apenas uma linha
```

- Comentários em múltiplas linhas (/* */)

```
/*
Exemplo de comentário no código onde mais de uma linha
pode ser utilizada
*/
```