



## Conceituação

"Um espaço na memória do computador, onde se pode guardar valores."

Existem 4 tipos:

- Instância: objeto
- Classe: classe
- Local: dentro de métodos
- Parâmetro: Na assinatura do método



## Criação

Convenções e regras:

- Não devem começar com números;
- Embora permitido, "\$" e "\_" devem ser evitados;
- São case-sensitive;
- Sem espaços;
- Não pode ser as palavra reservadas do Java:

```
abstract continue for new switch assert default goto package  
synchronized boolean do if private this break double case  
implements throw protected byte else import public throws  
enum instanceof return transient catch extends int short try  
char final interface static void class finally long strictfp  
volatile const float native super while
```

# Criação

## Convenções e regras:

- Exemplos:

- |                               |   |
|-------------------------------|---|
| ❖ <code>int i;</code>         | ❖ <code>int final j = 10;</code>                |
| ❖ <code>int l;</code>         | ❖ <code>int asrn24678md;</code>                 |
| ❖ <b><code>int 1a;</code></b> | ❖ <b><code>int asrn246 78md;</code></b>         |
| ❖ <code>int _1a;</code>       | ❖ <code>int asrn2\$4678_md = 10;</code>         |
| ❖ <code>int \$aq;</code>      | ❖ <b><code>int asrn2\$46%78_md = 10;</code></b> |
| ❖ <code>int l = 10;</code>    |   |

## Boas práticas

- Sempre começar com letra minúscula;
- Nomes expressivos;
- Notação camelo;
- Quando constante(final) maiúscula e separada por "\_";

## Boas práticas

- Exemplos:

- ❖ `int quantidadeProduto;`
- ❖ **`int QuantidadeProduto;`**
- ❖ `int final NUMERO_TENTATIVAS = 5;`
- ❖ **`int final numeroTentativas = 5;`**
- ❖ **`int NUMERO_TENTATIVAS = 5;`**
- ❖ `int qtdProd;`
- ❖ `int i;`

## Conceituação

"São os valores e consequentemente operações que as variáveis podem assumir e sofrer, respectivamente."

Tipificação:

- Estática(forte) vs Dinâmica(fraco)
- Primitivo vs Composto

## Conceituação

Opções de tipos:

- Textual
- Numeral
- Lógico
- Objeto

## Utilização

Exemplos numeral:

- byte: -128 até 127 → `byte b = 15;`
- short: -32.768 até 32.767 → `short s = -15785;`
- int: -2.147.483.648 até 2.147.483.647 → `int i = 8515785;`

## Utilização

Exemplos numeral:

- long: -9.223.372.036.854.775.808 até 9.223.372.036.854.775.807 → `long l = 5938515785L;`
- float:  $\pm 3.40282347E+38F$  → `float f = 3.14... (f);`
- double:  $\pm 1.79769313486231570E+308$  → `double d = 3.14... (d);`

# Utilização

Exemplos textual:

- **char:** caracteres de 16-bit unicode → `char c = '\u0084';` ou `char c = 'T';`
- **String:** um tipo "especial" → `String s = "T";`

# Utilização

Exemplos lógico:

- **boolean:** `true` e `false` → `boolean s = false;`

# Utilização

Exemplos objeto:

- Serão cenas dos próximos capítulos...

# Utilização

| Tipo de dado      | Valor default |
|-------------------|---------------|
| byte              | 0             |
| short             | 0             |
| int               | 0             |
| long              | 0L            |
| float             | 0.0f          |
| double            | 0.0d          |
| char              | '\u0000'      |
| String(e objetos) | null          |
| boolean           | false         |



# Boas práticas

Usar de forma adequada cada tipo de dado para cada informação

## Exercitando

Criar um simples projeto no IntelliJ e criar duas variáveis para cada tipo de dados apresentado.

Operadores

## Conceituação

"São símbolos especiais quais são capazes de realizar ações específicas em um, dois ou mais operandos e, em seguida, retornar um resultado."

## Conceituação

Tipos:

- pós-fixado: `exp++` ou `exp--`
- prefixado: `++exp` ou `--exp`
- aritmético: `+`, `-`, `*`, `/` e `%`
- atribuição: `=`, `+=`, `-=`, `*=`, `/=` e `%=`

## Utilização

Exemplos:

- |  |  |
|--|--|
| • <code>int i = ++k;</code> ➡ <code>i = k + 1;</code>        | • <code>double d = f;</code>                           |
| • <code>int j = k--;</code> ➡ <code>j = k; k = k - 1;</code> | • <code>i += 5;</code> ➡ <code>i = i + 5;</code>       |
| • <code>float f = 1.5f + 4.5f;</code>                        | • <code>j -= 3;</code> ➡ <code>j = j - 3;</code>       |
| • <code>long l = 10398L * 5L;</code>                         | • <code>d /= 2.7d;</code> ➡ <code>d = d / 2.7d;</code> |
| • <code>double d = 45d / 4d;</code>                          | • <code>l *= 3;</code> ➡ <code>l = l * 3;</code>       |
| • <code>int k = 15 % 4;</code>                               | • <code>k %= 2;</code> ➡ <code>k = k % 2;</code>       |

# Utilização

Precedências:

| Operador       | Precedência           |
|----------------|-----------------------|
| Pós-fixado     | exp++, exp--          |
| Prefixado      | ++exp, --exp          |
| Multiplicativo | *, /, %               |
| Aditivo        | +, -                  |
| Atribuição     | =, +=, -=, *=, /=, %= |

## Exercitando

Criar um simples projeto no IntelliJ e as variáveis e operações apresentadas. Crie expressões em que as precedências influenciem nos resultados.

Casting (conversão)

## Conceituação

"É a transformação de uma determinada variável de tipo menos específico para um tipo mais específico ou vice-versa."

## Conceituação

Tipos:

- Upcast(implícito)
- Downcast(explicito)

# Utilização

|        | byte  | short | char  | int   | long  | float | double |
|--------|-------|-------|-------|-------|-------|-------|--------|
| byte   |       | U - I | char  | U - I | U - I | U - I | U - I  |
| short  | D - E |       | char  | U - I | U - I | U - I | U - I  |
| char   | D - E | D - E |       | U - I | U - I | U - I | U - I  |
| int    | D - E | D - E | D - E |       | U - I | U - I | U - I  |
| long   | D - E | D - E | D - E | D - E |       | U - I | U - I  |
| float  | D - E | D - E | D - E | D - E | D - E |       | U - I  |
| double | D - E | D - E | D - E | D - E | D - E | D - E |        |

U - I : Upcast – Implícito    D - E: Downcast - Explícito

# Utilização

Exemplos:

- `long l; int i = 10; l = i;`
- `int i; long l = 100; i = (int) l;`
- `double d; float f = 10.5f; d = f;`
- `float f; double d = 10.5d; f = (float) d;`
- `int i; float f = 10.5f; i = (int) f;`

# Exercitando

Criar um simples projeto no IntelliJ e criar variáveis de vários tipos diferentes para assim realizar casting(conversões).

# Para saber mais

- <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html>
- <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>



DIGITAL  
INNOVATION  
ONE

## Para saber mais

- <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/keywords.html>
- <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>



DIGITAL  
INNOVATION  
ONE

## Para saber mais

- <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variablessummary.html>