



Fundamentos



Agenda

- Tipos Primitivos
- Palavras-Chave e Palavras Reservadas
- Constantes
- Operadores
- Conversão entre Tipos
- Exceções



Tipos Primitivos

- Fortemente tipada
- Oito (8) tipos primitivos
 - ♦ Seis (6) tipos primitivos numéricos
 - ♦ Quatro (4) tipos primitivos inteiros
 - ♦ Dois (2) tipos primitivos de ponto flutuante
 - ♦ Um (1) tipo primitivo caracter
 - ♦ Um (1) tipo primitivo booleano



Tipos Primitivos Numéricos

Tipos primitivos Inteiros

Tipo	Tamanho	Val. mínimo	Val. máximo
byte	1 byte	-128	127
short	2 bytes	-32.768	32.767
int	4 bytes	-2.147.483.648	2.147.483.647
long	8 bytes	-9.223.372.036.854.775.808	9.223.372.036.854.775.807



Literais Inteiros

- Podem ser expressos em formato decimal, octal ou hexadecimal
- O formato padrão é decimal

Ex.: **28**

- Para octal, o literal deve ser precedido por 0 (zero)

Ex.: **034**

- Para hexadecimal, o literal deve ser precedido por 0x ou 0X

Ex.: **0x1c** , **0x1C** , **0X1c** , **0X1C**

- Para binário, o literal deve ser precedido de 0b

Ex.: **0b0111_0001** , **0b010101**



Tipos Primitivos Numéricos

Tipos Primitivos de Ponto Flutuante

Tipo Tamanho		Intervalo
float	4 bytes	Aproximadamente +/- 3.40282347E+38F (6-7 dígitos significativos)
double	8 bytes	Aproximadamente +/- 1.79769313486231570E+308 (15 dígitos significativos)



Literais Ponto-Flutuante

- Para que um literal numérico seja interpretado como um valor em ponto-flutuante, este deve encaixar-se em um dos seguintes casos:

- Conter um ponto decimal

Ex.: **1.414**

- Conter a letra **E**, indicando notação científica

Ex.: **4.23E+21**

- Conter o sufixo **F** ou **f**, indicando um literal float

Ex.: **1.828f**

- Conter o sufixo **D** ou **d**, indicando um literal double

Ex.: **1234d**

- Um literal ponto-flutuante sem prefixo **F** ou **D** é interpretado como um literal double



Tipo Primitivo Caractere

char

- Utiliza-se apóstrofe para se representar constantes char
- Representa caracteres segundo o esquema Unicode (código de 2 bytes que permite a representação de 65.536 caracteres diferentes)

Ex:

```
char c = 'w';  
char c1 = '\u4567';
```




Tipo Primitivo Booleano

boolean

- É empregado em testes lógicos usando operadores relacionais que a linguagem Java suporta
- Valores possíveis:
 - true
 - false

Ex.: `boolean clienteEspecial = true;`
`boolean emDebito = false;`



Caracteres Especiais

Caracteres especiais

Seq. contr.	Nome	Valor Unicode
<code>\b</code>	backspace	<code>\u0008</code>
<code>\t</code>	tab	<code>\u0009</code>
<code>\n</code>	linefeed	<code>\u000a</code>
<code>\r</code>	carriage return	<code>\u000d</code>
<code>\"</code>	aspas	<code>\u0022</code>
<code>\'</code>	apóstrofe	<code>\u0027</code>
<code>\\</code>	barra invertida	<code>\u005c</code>



A Classe String

```
int it = 5;  
String st = String.valueOf(it);  
  
String sujeito = "Sócrates";  
String adjetivo = new String("filósofo");  
String nome = sujeito + " foi um grande ";  
nome += adjetivo;  
  
char[ ] texto = { 'J', 'a', 'v', 'a' };  
String titulo = new String(texto);
```



Variáveis

Palavras utilizadas para nomear variáveis, métodos e classes

- Devem ser iniciados por letra, dólar (\$), seguidos por zero ou mais letras, dólar ou dígitos
- Não podem ser palavras-chave ou palavras-reservadas
- Identificadores são sensíveis a letras maiúsculas ou minúsculas



Variáveis

- Ex. nomes válidos:

```
byte b21;  
int umaVariavelInteira;  
long $umaVariavelLong;  
char ch;
```

- Ex. nomes inválidos:

```
byte !bi;  
char @letra;  
double 43_variavel;
```



Palavras-Chave e Palavras Reservadas

abstract	long	super
assert	native	switch
boolean	new	synchronized
break	null	this
byte	package	throw
case	private	throws
catch	protected	transient
char	public	true
class	return	try
const	short	void
continue	static	volatile
default	strictfp	while
do		
double		
else		



Atribuições e Inicializações

```
int foo;  
foo = 37;
```

```
int i = 10;  
long l = 10, j = 20;
```

```
char charSim;  
charSim = 'S';
```



Constantes

- Em Java, usa-se a palavra-chave `final` para denotar uma constante
- `final` indica que pode ser atribuído valor a uma variável somente uma vez
- É costume dar nomes totalmente em maiúscula à constantes
- Ex.: `final double CM_POR_SOL = 2.54;`



Operadores

- São cinco (5) os tipos de operadores:
 - Atribuição
 - Aritmético
 - Operação de Bits
 - Relacionais
 - Lógicos



Operadores de Atribuição

- A expressão a direita é atribuída à variável a esquerda

```
int var1 = 0, var2 = 0;  
var1 = 50;           // var1 tem valor = 50  
var2 = var1 + 10;    // var2 tem valor = 60
```

- A expressão a direita é sempre executada antes da atribuição
- Atribuições podem ser agrupadas

```
int var3;  
var1 = var2 = var3 = 50;
```



Operadores Aritméticos

- Executa operações aritméticas básicas
- Trabalham com variáveis numéricas e literais

```
int a, b, c, d, e;  
a = 2 + 2; // adição  
b = a * 3; // multiplicação  
c = b - 2; // subtração  
d = c / 2; // divisão  
e = d % 2; // resto da divisão
```



Incremento e Decremento

- O operador ++ incrementa em 1
- O operador -- decrementa em 1

```
int var1 = 3;  
var1++;           // var1 agora é = 4
```



Incremento e Decremento

- O operador ++ pode ser usado de duas formas

```
int var2 = 3, var3 = 0, var4 = 0;
// pré-fixado:
//     primeiro incrementa var2
//     depois atribui o valor em var3
var3 = ++var2;
// pós-fixado:
//     primeiro atribui o valor em var4
//     depois decrementa var2
var4 = var2--;
```



Atribuição Combinada

- Um operador de atribuição pode ser combinado com qualquer operador aritmético

```
double total = 0, num = 1;  
double taxa = 0.5;
```

```
total = total + num;    // total é = 1  
total += num;           // total é = 2  
total -= 5;             // total é = -3  
total *= taxa;          // total é = -1.5
```



Conversões

As conversões de tipos podem ser divididas em 2 categorias:

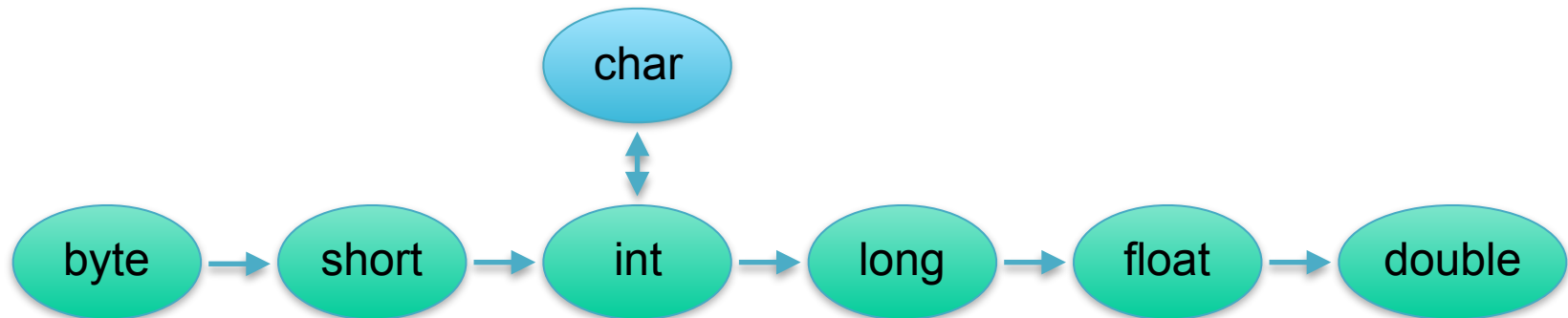
- Conversões Implícitas (Conversão)
- Conversões Explícitas (Cast ou Coerção)



Conversões

As regras gerais para conversão em atribuição de primitivas são:

- Um boolean não pode ser convertido para qualquer outro tipo.
- Um tipo não booleano pode ser convertido para outro tipo não booleano conforme o gráfico a seguir:





Conversões

Tipo a ser convertido	Converter em	Forma de conversão
<code>int x = 10;</code>	float	<code>float y = (float)x;</code>
<code>int x = 10;</code>	double	<code>double y = (double)x;</code>
<code>float x = 10.5f;</code>	int	<code>int y = (int)x</code>
<code>String x = "10";</code>	int	<code>int y = Integer.parseInt(x);</code>
<code>String x = "20.54";</code>	float	<code>float y = Float.parseFloat(x);</code>
<code>String x = "20.54";</code>	double	<code>double y = Double.parseDouble(x);</code>
<code>String x = "Java";</code>	Array de bytes	<code>byte[] y = x.getBytes();</code>
<code>int x = 10;</code>	String	<code>String y = String.valueOf(x);</code>
<code>float x = 10.5f;</code>	String	<code>String y = String.valueOf(x);</code>
<code>double x = 10.45;</code>	String	<code>String y = String.valueOf(x);</code>
<code>byte[] x = {'J', 'a', 'v', 'a'};</code>	String	<code>String y = new String(x);</code>



Wrappers

Tipo Primitivo

Classe Wrapper

boolean

java.lang.Boolean

char

java.lang.Character

byte

java.lang.Byte

short

java.lang.Short

int

java.lang.Integer

long

java.lang.Long

float

java.lang.Float

double

java.lang.Double