

JAVA. Identificadores

- Não podem começar por um dígito
- Podem ser constituídos por combinações de letras (língua Inglesa), dígitos e os caracteres `_` e `$`
- É usual respeitarem-se as seguintes convenções:
 - Nome de uma **classe** começa por maiúscula (e.g. `Solido`)
 - Nome de um **subprograma** começa por minúscula (e.g. `main()`)
 - Nome de uma **variável** começa por minúscula (e.g. `volume`)
 - Nome de uma **constante** é escrito em maiúsculas (`MAX_VEC`)

JAVA. Tipos Primitivos

TIPO	VALORES	VALOR OMISSÃO	#BITS	GAMA DE VALORES
<i>boolean</i>	true false	false	1	
<i>char</i>	Unicode (compat. ASCII)	\u0000	16	\u0000 a \uFFFF
<i>byte</i>	Inteiro c/ sinal	0	8	-128 a 127
<i>short</i>	Inteiro c/ sinal	0	16	-32768 a 32767
<i>int</i>	Inteiro c/ sinal	0	32	-2147483648 a 2147483647
<i>long</i>	Inteiro c/ sinal	0	64	-9223372036854775808 a 9223372036854775807
<i>float</i>	IEEE 754 FP	0.0	32	$\approx \pm 3.4E+38$ a $\pm 1.4E-45$
<i>double</i>	IEEE 754 FP	0.0	64	$\approx \pm 1.8E+308$ a $\pm 5E-324$

JAVA.DeclaraçãoDeVariáveis

```
id_tipo id_variável1 [= valor1] [, id_variável2 [= valor2] ...];
```



```
int x;  
  
int x = 10; /* declaração com inicialização */  
  
int x = 20, y, z = 30;  
  
int x, y = 10;  
  
int a = x+ y;  
  
  
char um = '1';  
  
char c = 'A'; /* formato UNICODE, caracteres ASCII compatíveis */  
char newline = '\n';  
  
boolean fim;  
  
boolean fechado = true;  
  
  
byte b1 = 0x49 ; /* hexadecimal * /  
  
  
long diametro;  
  
long raio = -1.7E+5;  
  
  
double d;  
  
double small$123 = .0000000123;  
  
double pi = 3.14159273269;
```

JAVA.DeclaraçãoDeConstantes

- semelhante à declaração de variáveis
- acrescida do atributo *final*
- obrigatório indicar o valor da constante



```
final float PI = 3.14159273269;
```

```
final float VLV = 2.99792458E8; //Velocidade da luz no vácuo em m/s
```

```
final float AG = 9.80665; //aceleração da gravidade em m/s^2
```

```
final float ME = 9.109389E-31; //massa electrão repouso em kg
```

JAVA. Conversão Entre Tipos

- O tipo do resultado de uma expressão aritmética depende do tipo dos operandos
- Muitas expressões contêm valores de vários tipos
- Os operadores aritméticos estão definidos para funcionar com operandos do mesmo tipo

∴ O computador terá que fazer conversões de tipo automaticamente e de modo a que não haja perda de informação

- Nem todas as transformações são possíveis
 - Diferente tipo de representação (e.g. valor real para um tipo inteiro???)
 - Diferença no espaço de memória ocupado pelos valores dos diversos tipos (e.g. um valor `long` num valor `int`)

∴ É possível converter um valor para um tipo que ocupe mais espaço, mas o inverso não é verdadeiro

`byte > short > int > long > float > double`



```
int var_int = 10; // resultado_int;

double var_double = 5.2; // resultado_double;

double resultado_double = var_int + var_double;

int resultado_int = var_int + var_double;

/* erro de compilação " Incompatible type for =. Explicit cast
needed to convert double to int".*/
```

- A conversão de tipos com perda de informação é permitida utilizando de forma explícita o operador de coerção (*cast*) - tipo pretendido entre parêntesis antes da expressão a converter



```
resultado_int = (int) (var_int + var_double);

char c1, c2; char c3 = 'a';

int x = 67; int y = 4;

c1 = (char) (x + y) ;

c2 = (char) ( c3 + 1 );
```

JAVA.Operadores

Pr.	OPERADOR	OPERANDO(S)	ASSOC.	DESCRIÇÃO
1	++ --	Aritméticos	D	Pré pós in de cremento
	+ -	Aritmético	D	Sinal
	~	Byte, short, char, int, long (integral)	D	Complemento
	!	Booleano	D	Negação
	(<i>tipo</i>)	Qualquer	D	Conversão
2	* / %	Aritméticos	E	Multiplicação, divisão e resto
3	+ -	Aritméticos	E	Soma e subtracção
	+	Strings	E	Concatenação
4	<< >> >>>	Integral	E	Deslocamento de bits esq., dir. direita com 0
5	< <= > >=	Aritméticos	E	Relacionais
6	instanceof	Objecto, tipo	E	Teste de tipo
	== !=	Primitivos	E	Valor igual e valor diferente
	== !=	Objectos	E	Objectos iguais e diferentes
7	&	Integral	E	E de bits
	&	Booleano	E	E lógico
8	^	Integral	E	OU exc. de bits
	^	Booleano	E	OU exc. lógico
9		Integral	E	OU de bits
		Booleano	E	OU lógico
10	&&	Booleano	E	E condicional
11		Booleano	E	OU condicional
12	?:	Booleano, qq, qq	E	Alternativa
13	= *= /=	Variável, qualquer	D	Atribuição e atribuição com operação
	%= += -=			
	<<= >>= >>>=			
	&= ^= =			

