Universidade de Mogi das Cruzes
Implementação Orientada a Objetos - Profª. Danielle Martin

Guia da Sintaxe do Java

TIPOS PRIMITIVOS DE DADOS DO JAVA

São os tipos nativos de dados do Java, que podem ser usados na declaração de atributos, variáveis, parâmetros.

Tipo primitivo	Tamanho	Valor padrão	Alcance
boolean	1 bit	false	false ou true
char	16 bits	'\u0000'	caracteres unicode
byte	8 bits	0	-128 a 127
short	16 bits	0	-32768 a 32767
int	32 bits	0	-2147483648 a 2147483647
long	64 bits	0	-9223372036854775808 a 9223372036854775807
float	32 bits	0.0f	+/-3.4E-38 a +/- 3.4E+38
double	64 bits	0.0d	+/-1.7E-308 a +/-1.7E+308
instâncias	-	null	-

NOMENCLATURA DE IDENTIFICADORES

Identificadores em Java (nomes de classes, variáveis, atributos, métodos, parâmetros), podem iniciar apenas com:

- letra
- um sublinhado (_)
- um símbolo de dólar ($)

Boas práticas de programação OO sugerem para a nomenclatura de identificadores:

- Classes - a primeira letra de cada palavra em maiúscula. Ex. NomeDaClasse
- Atributos, variáveis e métodos - lowerCamelCase. Ex. nomeDoAtributo, meuMetodo, boasPraticas

Exemplos:

```
int  idade = anoAtual - anoNascimento;
double salarioGerente;
String nome;
```

OPERADORES

Operadores de Atribuição		
=	x = 1	Atribui o valor 2 a x
+=	x += 2	Equivale a x = x + 2
-=	x -= 3	Equivale a x = x - 3
*=	x *= 4	Equivale a x = x * 4
/=	x /= 5	Equivale a x = x / 5
%=	x %= 6	Equivale a x = x % 6

Operadores Aritméticos		
+	x + y	soma (ou concatena) valores de x e y
-	x - y	subtrai o valor de y do valor de x
*	x * y	multiplica o valor de x pelo valor de y
/	x / y	divide o valor de x pelo valor de y
%	x % y	retorna o resto da divisão do valor de x pelo valor de y
++	x++	incrementa o valor de x em 1
--	x--	decrementa o valor de x em 1

Operadores Relacionais		
<	a < b	retorna true se a for menor que b
<=	a <= b	retorna true se a for menor ou igual a b
>	a > b	retorna true se a for maior que b
>=	a >= b	retorna true se a for maior ou igual a b
==	a == b	retorna true se a for igual a b
!=	a != b	retorna true se a for diferente de b

Operadores Lógicos (somente operam sobre valores booleanos)		
&	a & b	AND (E) - retorna true se a E b forem true
&&	a && b	AND (E) com curto circuito. Ex: if (x!=0 && y/x == 1) // nunca irá causar erro de execução
|	a | b	OR (OU) - retorna true se a OU b forem true

ll	a ll b	OR (OU) com curto circuito. Ex: if (x==0 ll y/x == 1) // nunca irá causar erro de execução
^	a ^ b	XOR (OU Exclusivo) - retorna true se a OU b forem true, mas nunca ambos
!	!a	NOT (converte true em false e vice versa)

Operador Condicional (ou ternário)		

?:	x = (b) ? 1 : 2	Se b for true, atribui o valor 1 a x, senão atribui o valor 2 a x.
		Equivale a:
		if (b) {
		  x = 1;
		} else {
		  x = 2;
		}

PALAVRAS RESERVADAS

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

CONVERSÃO DE TIPOS

Implícita - de tipos menores para tipos maiores.
Ex.

```
int a = 5;
float b = a;
```

Explícita (Casting) - conversão de tipos maiores para tipos menores, pode haver perdas.
Ex.

```
double c = 5.95;
int d = (int) c;

//d será igual a 5
```

COMENTÁRIOS

Comentários de uma linha: //meu comentario
Comentários de múltiplas linhas /* meu comentario */
Comentários javadoc /** documentacao do codigo */

ESTRUTURAS CONDICIONAIS

```
if (condicao) {
  /* bloco que executa se
     condição for verdadeira
   */
}
```

```
if (condicao) {
   /* bloco que executa se
     condição for verdadeira
   */
} else {
   /* bloco que executa se
     condição for falsa
   */
}
```

```
if (condicao) {
  /* bloco que executa se
     condição for
     verdadeira
   */
} else if (condicao2) {
  /* bloco que executa se
     condição2 for
     verdadeira
   */
} else {
  /* bloco que executa se
     nenhuma das condições
     forem verdadeiras
   */
}
```

```
int i = 5;
switch (i) {
   case 5:
      System.out.println(“Mensagem 1”);
      break;
   case 4:
      System.out.println(“Mensagem 2”);
      break;
   default:
      System.out.println(“Mensagem 3”);
}
```

ESTRUTURAS DE REPETIÇÃO

```
int i = 0;

do {
    System.out.println(i);
    i++;
} while (i<5)
```

```
int i = 0;

while (i<5) {
    System.out.println(i);
    i++;
}
```

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

```
int[] vetor = {1, 2, 3, 5, 7, 11}

for (int elemento : vetor) {
   System.out.println(elemento);
}
```

```
String[] vetor = new String[5];
for (int i = 0; i<5; i++) {
     System.out.println("Digite um texto ");
     vetor[i] = teclado.next();
     System.out.println(vetor[i]);
}
```

SAÍDA DE DADOS (CONSOLE)

```
//Imprime no console
System.out.println("Minha mensagem");
```

ENTRADA DE DADOS (CONSOLE)

```
//Cria um novo scanner do teclado
Scanner teclado = new Scanner(System.in);

System.out.println("Digite um texto: ");
String b = teclado.next(); // lê uma string do console

System.out.println("Digite um número inteiro: ");
int a = teclado.nextInt();  // lê um int do console

System.out.println("Digite um número real: ");
float b = teclado.nextFloat();  // lê um float do console
```